UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE COMUNICACIONES

DESIGN, DEVELOPMENT AND EVALUATION OF AN ADAPTIVE AND STANDARDIZED RTP/RTCP-BASED IDMS SOLUTION

by

MARIO MONTAGUD CLIMENT

Advisors:

PROF. DR. FERNANDO BORONAT SEGUI (UPV)

DR. PABLO CESAR (CWI)

March 2015

ABSTRACT

Nowadays, we are witnessing a transition from physical togetherness towards networked togetherness around media content. Novel forms of shared media experiences are gaining momentum, allowing geographically distributed users to concurrently consume the same media content while socially interacting (e.g., via text, audio or video chat). Relevant use cases are, for example, Social TV, networked games and multi-party conferencing.

However, realizing enjoyable shared media services faces many challenges. In particular, a key technological enabler is the concurrent synchronization of the media playout across multiple locations, which is known as Inter-Destination Multimedia Synchronization (IDMS).

This PhD thesis presents an inter-operable, adaptive and accurate IDMS solution, based on extending the capabilities of RTP/RTCP standard protocols (RFC 3550). Concretely, two new RTCP messages for IDMS have been defined to carry out the necessary information to achieve IDMS. Such RTCP extensions have been standardized within the IETF, in RFC 7272. In addition, novel standard-compliant Early Event-Driven (EED) RTCP feedback reporting mechanisms have been also designed to enhance the performance in terms of interactivity, flexibility, dynamism and accuracy when performing IDMS.

The designed IDMS solution makes use of globally synchronized clocks (e.g., using NTP) and can adopt different (centralized and distributed) architectural schemes to exchange the RTCP messages for IDMS. This allows efficiently providing IDMS in a variety of networked scenarios and applications, with different requirements (e.g., interactivity, scalability, robustness…) and available resources (e.g., bandwidth, latency, multicast support…). Likewise, various monitoring and control algorithms, such as dynamic strategies for selecting the reference timing to synchronize with, and fault tolerance mechanisms, have been added. Moreover, the proposed IDMS solution includes a novel Adaptive Media Playout (AMP) technique, which aims to smoothly adjust the media playout rate, within perceptually tolerable ranges, every time allowable asynchrony thresholds are exceeded.

Prototypes of the IDMS solution have been implemented in both a simulation and in real media framework. The evaluation tests prove the consistent behavior and

the satisfactory performance of each one of the designed components (e.g., protocols, architectural schemes, master selection policies, adjustment techniques…). Likewise, comparison results between the different developed alternatives for such components are also provided. In general, the obtained results demonstrate the ability of this RTP/RTCP-based IDMS solution to concurrently and independently maintain an overall synchronization status (within allowable limits) in different logical groups of users, while avoiding annoying playout discontinuities and hardly increasing the computational and traffic load.

RESUMEN

Hoy en día, estamos asistiendo a un cambio de paradigma en cuanto al consumo de contenidos multimedia. Nuevas experiencias multimedia compartidas están cobrando impulso, permitiendo el consumo simultáneo de contenidos multimedia por parte de múltiples usuarios distribuidos en red, a la vez que interactúan mediante servicios de chat (ya sea texto, audio o video). Casos de uso relevantes son, por ejemplo, la TV Social, juegos en red multi-jugador o servicios de audio/video conferencia en grupo.

Sin embargo, proporcionar de manera satisfactoria dichos servicios multimedia compartidos supone una serie de desafíos. En particular, un reto clave es conseguir la sincronización simultánea de los procesos de reproducción en cada uno de los receptores involucrados, lo que se conoce como Sincronización Multimedia Inter-Destinatario (Inter-Destination Multimedia Synchronization, IDMS).

En esta Tesis se presenta una solución de IDMS inter-operable, adaptativa y precisa, basada en la extensión de las funcionalidades de los protocolos estándar RTP/RTCP (RFC 3550). En concreto, dos nuevos mensajes RTCP se han definido para intercambiar información necesaria para conseguir IDMS. Dichas extensiones del protocolo RTCP se han estandarizado en el seno del IETF, en la RFC 7272. Además, se han diseñado mecanismos novedosos, aunque compatibles con los estándares existentes, para el envío de mensajes RTCP de manera inmediata y basada en eventos, con el objetivo de mejorar las prestaciones en cuanto a interactividad, flexibilidad, dinamismo y precisión en servicios multimedia que requieren IDMS.

La solución de IDMS diseñada se basa en el uso de relojes globales (p.ej., utilizando NTP) y puede adoptar diferentes esquemas arquitecturales (centralizados y distribuidos) para intercambiar los mensajes RTCP definidos. Esto permite proporcionar IDMS de manera eficiente en un gran variedad de escenarios y aplicaciones, con distintos requisitos (p.ej., interactividad, escalabilidad, robustez…) y recursos disponibles (p.ej., ancho de banda, retardos, soporte de multicast…). Asimismo, también se han incorporado varios algoritmos de monitorización y control, tales como estrategias dinámicas para la selección de la referencia maestra para la sincronización, así como mecanismos de tolerancia a fallos. Además, se ha diseñado una técnica novedosa de ajuste suavizado de la tasa

de reproducción (Adaptive Media Playout, AMP), dentro de rangos tolerables, cada vez que se detectan asincronías superiores a umbrales pre-establecidos.

Se han implementado prototipos de la solución de IDMS tanto en una plataforma de simulación como en una real. Las pruebas de evaluación muestran el comportamiento consistente y el rendimiento satisfactorio de cada unos de los componentes diseñados (p.ej., protocolos, esquemas arquitecturales, políticas de selección de la referencia maestra, técnicas de ajuste…). Asimismo, se proporcionan resultados comparativos para las diferentes alternativas de cada uno de dichos componentes. En general, los resultados obtenidos demuestran la capacidad de la solución de IDMS de mantener, de manera simultánea e independiente, un estado de sincronización global (por debajo de límites permisibles) en differentes grupos lógicos de usuarios, al mismo tiempo que se minimizan discontinuidades molestas en los procesos de reproducción y apenas incrementando la sobrecarga de tráfico de red y computacional.

RESUM

Avui en dia, estem assistint a un canvi de paradigma pel que fa al consum de continguts multimèdia. Noves experiències multimèdia compartides estan cobrant impuls, permetent el consum simultani de continguts multimèdia per part de múltiples usuaris distribuïts en xarxa, alhora que interactuen mitjançant serveis de xat (ja siga text, àudio o vídeo). Casos d'ús rellevants són, per exemple, la TV Social, jocs en xarxa multi-jugador o serveis d'audio/video conferència en grup.

No obstant això, proporcionar de manera satisfactòria aquests serveis multimèdia compartits suposa una sèrie de desafiaments. En particular, un repte clau és aconseguir la sincronització simultània dels processos de reproducció en cadascun dels receptors involucrats, el que es coneix com a Sincronització Multimèdia Inter-Destinatari (Inter-Destination Multimedia Synchronization, IDMS).

En aquesta Tesi es presenta una solució d'IDMS inter-operable, adaptativa i precisa, basada en l'extensió de les funcionalitats dels protocols estàndard RTP/RTCP (RFC 3550). En concret, dos nous missatges RTCP s'han definit per a intercanviar informació necessària per aconseguir IDMS. Aquestes extensions del protocol RTCP s'han estanditzat en el si de l'IETF, en la RFC 7272. A més, s'han dissenyat mecanismes innovadors, encara que compatibles amb els estàndards existents, per a l'enviament de missatges RTCP de manera immediata i basada en esdeveniments, amb l'objectiu de millorar les prestacions en quant a interactivitat, flexibilitat, dinamisme i precisió en serveis multimèdia que requereixen IDMS.

La solució d'IDMS dissenyada es basa en l'ús de rellotges globals (p.ex., utilitzant NTP) i pot adoptar diferents esquemes arquitecturals (centralitzats i distribuïts) per a intercanviar els missatges RTCP definits per a IDMS. Això permet proporcionar IDMS de manera eficient en un gran varietat d'escenaris i aplicacions, amb diferents requisits (p.ex., interactivitat, escalabilitat, robustesa...) i recursos disponibles (p.ex., ample de banda, retards, suport de multicast...). Així mateix, diversos algorismes de monitorització i control, com ara estratègies dinàmiques per a la selecció de la referència mestra per a la sincronització i mecanismes de tolerància a fallades, s'han afegit. A més, s'ha dissenyat una tècnica innovadora d'ajust suavitzat de la velocitat de reproducció (Adaptive Media Playout, AMP), dintre de marges tolerables, cada vegada que es detecten asincronies superiors a límits pre-establerts.

S'han implementat prototips de la solució d'IDMS, tant en una plataforma de simulació com en una real. Les proves d'avaluació mostren el comportament consistent i el rendiment satisfactori de cadascuns dels components dissenyats (p.ex., protocols, esquemes arquitecturals, polítiques de selecció de la referència mestra, tècniques d'ajust...). Així mateix, es proporcionen resultats comparatius per a les diferents alternatives de cadascun dels components dissenyats. En general, els resultats obtinguts demostren la capacitat de la solució d'IDMS per a mantenir, de manera simultània i independent, un estat de sincronització global (per sota de límits permissibles) en diferents grups lògics d'usuaris, a la vegada que es minimitzen discontinuïtats molestes en els processos de reproducció, així com incrementant molt poc la sobrecàrrega de tràfic de xarxa i computacional.

ACKNOWLEDGMENTS

I would like to spend a few words to thank all the people who have supported, helped and guided me during this four-year stage.

First and foremost, my deepest gratitude goes to my PhD advisor Prof. Fernando Boronat. Fernando was the first one who trusted in me, and he opened me the door to the research world. Thanks to Fernando, I have been able to actively participate in many projects and to attend many research events all over the world. This has allowed me to meet many people and to learn a lot. Fernando has always been there, irrespective of the time and irrespective of the problem, and his familiarity, timely feedback and suggestions have made things much easier.

Parallel to Fernando, I would also like to thank the people from UPV who have supported and helped me during these years, including professors (V.Vidal, B.Roig, A.Sapena, V.Gregori, J.V. Morro, J.L.Corral, C.Palau…), PhD colleagues (M.Garcia, D. Bri…) and colleagues from our emerging research group (J.Pastor, J.Belda, M.Martínez…).

Second, I am also immensely thankful to my co-supervisor Dr. Pablo Cesar. He has continuously supported me during these last years. His vision and advices have been key for the successful completion of this PhD thesis. Thanks to Pablo, I have met many distinguished researchers, both from CWI (Prof. Dick Bulterman, Jack Jansen…) and from other places all over the world. Many thanks also go to all the members of the DIS group at CWI, who have tried hard to make me feel welcome during my stays there.

Third, I would like to thank the researchers from TNO with whom I have collaborated during these years, especially to Hans Stokking. Their feedback, broad knowledge and suggestions about the applicability of the research carried out within the context of this PhD thesis have been very valuable and appreciated.

Fourth, I would also like to thank all the members of the evaluation committee of this PhD thesis for their collaboration and constructive feedback, as well as Prof. Klara Nahrstedt for her valuable comments and suggestions since the Doctoral Symposium at ACM International Conference on Multimedia 2013.

Fifth, I would also like to express my gratitude to the two entities that have made possible this PhD thesis. On the one hand, UPV granted me with a four-year research scholarship and with a three-month stay at CWI. On the other hand, CWI granted me with a three-month scholarship and with a one-year postdoc contract there.

Last, but not least, I want to extend my thanks to my family, especially to my brothers and parents. They continuously give me encouragement, support and love. This PhD thesis is dedicated to them.

TABLE OF CONTENTS

LIST OF ACRONYMS

3DTI	3D Tele-Immersion
3GPP	3rd Generation Partnership Project
ACM	Association for Computing Machinery
ACT	(RTCP APP) Action Packet
AL-FEC	Application Layer Forward Error Correction
AMP	Adaptive Media Playout
APP	(RTCP) Application-Defined Packet
ASM	Any Source Multicast
AU	Access Unit
AvgD	Average Delay
AvgP	Average Playout Rate
AVP	Audio-Visual Profile
AVPF	Audio-Visual Profile with Feedback
AVTCORE	Audio Video Transport Core
BT	Block Type
C-to-C	Cluster-to-Cluster
CBR	Constant Bit Rate
CDF	Cumulative Distribution Function
CDN	Content Delivery Network
CMTS	Cable Modem Termination System
CNAME	Canonical Name
CoD	Content on Demand

CPU	Central Processing Unit
CVE	Collaborative Virtual Environment
DASH	(MPEG) Dynamic Adaptive Streaming over HTTP
DCR	Degradation Category Rating
DCS	Distributed Control Scheme
DiffServ	Differentiated Services
DL	Digital Library
DS	Distribution Source
DSLAM	Digital Subscriber Line Access Multiplexer
DTS	Decoding Time Stamp
DTV	Digital TV
DVB	Digital Video Broadcasting
EED	Early Event-Driven
EPG	Electronic Program Guide
ETSI	European Telecommunications Standards Institute
FCI	Feedback Control Information
FIFO	First-In First-Out
FMT	Frame Message Type
FPS	Frames Per Second
FT	Feedback Target
FTP	File Transfer Protocol
FTTH	Fiber To The Home
GAL	GALILEO
GLONASS	Global Navigation Satellite System
GoP	Group of Pictures
GPS	Global Positioning System
GS	Group Synchronization
GTD	Game Time Difference
HbbTV	Hybrid Broadcast Broadband TV

HD	High Definition
HDS	HTTP Dynamic Streaming
HLS	HTTP Live Streaming
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IDES	Inter-Device Synchronization
IDMS	Inter-Destination Multimedia Synchronization
IETF	Internet Engineering Task Force
ILA	Interactivity Loss Avoidance
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPTV	Internet Protocol Television
ISO	IDMS Session Object
ISP	Internet Service Provider
IU	Information Unit
ITU	International Telecommunication Union
LAN	Local Area Network
LDU	Logical Data Unit
MAC	Media Access Control
M/S	Master/Slave
MCU	Multipoint Conference Unit
MDC	Multiple Description Coding
MDF	Media Distribution Function
MdP	Mean Discrepancy From the Original Playout Rate
MDU	Media Data Unit
MOG	Multi-player Online Games
MOS	Mean Opinion Score
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group

MPEG-TS	MPEG Transport Stream
MS-SSTR	Microsoft Smooth Streaming Protocol
MSAS	Media Synchronization Application Server
MSE	Mean Square Error
MSP	Multipoint Synchronization Protocol
MSS	Maximum Segment Size
MU	Media Unit
NAT	Network Address Translation
NCL	Nested Context Language
NGN	Next Generation Network
NS-2	Network Simulator 2
NTP	Network Time Protocol
OIPF	Open IPTV Forum
OS	Operating System
OSI	Open System Interconnection
OTT	Over-The-Top
P2P	Peer-to-Peer
PAL	Phase Alternation Line
PDF	Probability Distribution Function
PER	Packet Error Rate
PESQ	Perceptual Evaluation of Speech Quality
PPM	Parts Per Million
PSI	Program Specific Information
PSNR	Peak Signal to Noise Ratio
PT	(RTCP) Packet Type
PT	(RTP) Payload Type
PTP	Precision Time Protocol
PTS	Presentation Time Stamp
PVR	Personal Video Recorder

QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Point
RED	Random Early Detection
RFC	Request For Comments
RMSE	Root Mean Square Error
RQ	Research Question
RR	(RTCP) Receiver Report
RTCP	RTP Control Protocol
RTP	Real-Time Transport Protocol
RTSP	Real Time Streaming Protocol
RTT	Round Trip Time
SCF	Service Control Function
SD	Standard Definition
SDES	(RTCP) Source Description
SDP	Session Description Protocol
SECAM	Séquentiel Couleur Avec Mémoire
SIP	Session Initiation Protocol
SMIL	Synchronized Multimedia Integration Language
SMS	Synchronization Maestro Scheme
sMS	synchronous Media Sharing
SPST	Synchronization Packet Sender Type
SR	(RTCP) Sender Report
SSIM	Structural Similarity
SSM	Single Source Multicast
SSRC	Synchronization Source
STB	Set Top Box
StdP	Standard Deviation of the Playout Rate
SVC	Scalable Video Coding

SyncGroupId	Synchronization Group Identifier
SC	Sync Client
TAI	International Atomic Time
TISPAN	Telecoms & Internet Converged Services & Protocols for Advanced Networking
Tcl	Tool Command Language
TCP	Transmission Control Protocol
tob	Buffer Time Offset
ToS	Type of Service
TS	Transport Stream
TSS	Trailing State Synchronization
TV	Television
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
VBR	Variable Bit Rate
VDSL2	Very-High-Bit-Rate Digital Subscriber Line 2
VLC	Video LAN Client
VoD	Video on Demand
VoIP	Voice over IP
VTR	Virtual Time Rendering
VQMT	Video Quality Measurement Tool
VQM	Video Quality Metric
WAN	Wide Area Network
WebRTC	Web Real-Time Communication
WG	Working Group
W3C	World Wide Web Consortium
WLAN	Wireless LAN
XR	(RTCP) Extended Report

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

The following Table lists the symbols defined or used in this PhD thesis, their meaning, and the Chapter in which they firstly appear.

Table of Symbols. List of Symbols defined or used in this PhD thesis

Symbol	Units	Meaning	Chapter
θ	MU/s	Media Server Nominal Rate	Chapter 2
γ^k_i	%	Playout Rate Skew (Deviation) of the *i-th* Sync Client belonging to the *k-th* Group	Chapter 2
$\omega^k_i(t)$	%	Playout Rate Drift (Fluctuation Rate) of the *i-th* Sync Client belonging to the *k-th* Group	Chapter 2
ε	%	Maximum Value of Playout Rate Drift	Chapter 2
$\mu^k_i(t)$	MU/s	Playout Rate of the *i-th* Sync Client belonging to the *k-th* Group	Chapter 2
t_{ini}	Time Unit	Initial transmission instant	Chapter 2
b_{ini}	s	Initial buffering delay	Chapter 2
$A(t,\gamma)$	MUs	Asynchrony between Sync Clients, according to the temporal evolution (t) of the session and their playout rate deviations (γ).	Chapter 2
t_n	Time Unit	Transmission time of the *n-th* MU	Chapter 7
$r^k_{n,i}$	Time Unit	Reception time of the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group	Chapter 7
$v_{n-1,n}$	Timestamp Units	Delay variation for two consecutive *(n-1)-st* and *n-th* packets	Chapter 7
j_n	Timestamp Units	Jitter for the *n-th* packet	Chapter 7
$BW_{session}$	kbps	Bandwidth of the RTP Session	Chapter 7
$RTCP_{size}$	bits	Average size of all sent and received RTCP packets	Chapter 7
T^{3550}_{RTCP}	s	RTCP Report Interval according to RFC 3550	Chapter 7
$T^{3550}_{RTCP\_d\_min}$	s	Minimum RTCP Report Interval according to RFC 3550	Chapter 7
T^{4585}_{RTCP}	s	RTCP Report Interval according to RFC 4585	Chapter 7
trr-int	s	Offset to the RTCP Report Interval (RFC 4585)	Chapter 7

t'_n	Timestamp Units	RTP timestamp of the *n-th* MU	Chapter 8
$p^k_{n,i}$	Time Unit	Presentation or playout time of the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group	Chapter 8
$b^k_{n,i}$	ms	Buffering delay of the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group	Chapter 8
$d^k_{n,i}$	ms	Playout or end-to-end delay of the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group	Chapter 8
τ^k_{max}	ms	Maximum allowable asynchrony threshold in the *k-th* Group	Chapter 8
$\Delta^k_{n,i}$	ms	Detected asynchrony for the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group	Chapter 8
p_{ini}	Time Unit	Initial playout instant	Chapter 8
$s^k_{n,\,i}$	ms	Service time for the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group	Chapter 8
$\delta^k_{n,i}$	ms	Distributed asynchrony for the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group	Chapter 8
$\varphi^k_{n,i}$	%	Playout Factor for the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group	Chapter 8
N^{AMP}	-	Number of MUs involved in the AMP process	Chapter 8
$t_{d(n)}$	Time Unit	*n-th* Scheduled (Deterministic) RTCP Transmission Time	Chapter 9
$t_{r(n)}$	Time Unit	*n-th* Real (Randomized) RTCP Transmission Time	Chapter 9
DD	ms	(Event) Detection Delay	Chapter 9
AD	ms	(Event) Adjustment Delay	Chapter 9
Δt_1	ms	Delay since a latecomer joins an on-going session and it can send an RTCP-IDMS-REQ message	Chapter 9
Δt_2	ms	Sync Manager Delay	Chapter 9
Δt_3	ms	Playout Adjustment Delay	Chapter 9
$l_{n,i}$	ms	Latency or network delay for the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group	Chapter 10

Chapter 1

INTRODUCTION

1.1 Context of the PhD thesis

Certain traditional forms of media consumption involve social interaction between groups of users. For instance, people often gather at a single location for consuming media (e.g., for watching TV content) together. The typical scenario is a group of friends watching a live football match at a friend's home. Similarly, people often invite friends or family members at home to show them videos or photos from their holidays or celebrations. In such scenarios, multiple co-located people socially interact within the context of specific media content consumption (typically consumed on a common device). Actually, the shared consumption of media content is frequently the catalyst why the users meet up, as it allows socializing, discussing about common interests, re-living shared (past) experiences/memories, sharing emotions, and increasing the users' engagement ([Wij12a], [Tim14]), thus contributing to strength the social bonding.

Unfortunately, many times, a myriad of practical issues prevent people from physically meeting up. The world has become a global society, and people move to different geographical locations (cities, countries or continents) for study, vacations, job, business, among other purposes. In spite of the geographical segregation, people are still interested in remaining connected and in socializing with their relatives and friends. Therefore, the need for recreating such shared media experiences, while apart, has arisen.

This transition from physical togetherness towards networked togetherness around media content is becoming a reality thanks to the latest advances on media delivery technologies and on social networking, in conjunction with the proliferation of connected devices. Novel forms of shared media experiences are gaining momentum, allowing geographically distributed users to socially interact (e.g., via text, audio or video chat, or combinations thereof) within the context of simultaneous content consumption. Relevant use cases are Social TV, networked

games, multi-party conferencing, synchronous e-learning and collaborative tele-work, just to cite a few of them. For instance, the co-located friends in the above example can now watch the football match from their own home, while being able to converse, discuss about its evolution, and cheer together when goals are scored.

This emerging media consumption paradigm opens the door to a range of new possibilities and emerging business models (e.g., it also allows saving time and costs, by preventing from traveling in various use cases, such as in e-learning and in e-meetings).

However, realizing enjoyable shared interactive services faces many technological (e.g., Quality of Service or QoS, design of efficient media adaptation and delivery methods, cross-domain session handling, integration of interaction channels, scalability, synchronization, etc.) and perceptual (e.g., presence awareness, privacy concerns, Quality of Experience or QoE) challenges [Vai11b].

This PhD thesis focuses on a key technological enabler, which is drawing the attention of academy and industry alike: the concurrent synchronization of the media playout across the involved (geographically distributed) devices. This process is known as Inter-Destination Media Synchronization (IDMS). In absence of IDMS, the interactions between the users in shared media experiences will not be consistent. For instance, in the above *"watching apart together"* scenario, being aware of a goal through the cheering of a friend via the chat channel, before the goal sequence is displayed on the local screen, can be very frustrating and would spoil the shared experience [Mek12].

In particular, this PhD thesis aims to explore the IDMS use cases and their associated challenges, to study the work that has been done in this area, and to derive the requirements and components needed to efficiently provide IDMS in a variety of scenarios and distributed media applications. It also provides an exhaustive overview of the necessary components to enable IDMS, the required cooperation between them, and a discussion about the suitability and feasibility of different alternatives for several of such components. Most importantly, the key contribution of this PhD thesis is the design, development and evaluation of an inter-operable, adaptive and accurate IDMS solution, fitting the identified requirements of the emerging distributed media consumption paradigm. This IDMS solution is based on the extension of the capabilities of the Real-Time Transport (RTP) and RTP Control Protocol (RTCP) standard protocols (specified in RFC 3550), and integrates different architectural schemes, control mechanisms and adjustment techniques. A key point of the presented RTP/RTCP-based IDMS solution is that it is an evolutionary approach, being backwards compatible with existing standard media delivery technologies and systems. Actually, the specification of the RTCP extensions to achieve IDMS has been standardized within the umbrella of the Internet Engineering Task Force (IETF), in RFC 7272.

1.2 Research Goals

The **main goal** of this PhD thesis is to *design, develop and evaluate an inter-operable, adaptive and accurate IDMS solution*.

However, in order to constitute a complete IDMS solution, the integration and/or interaction between many components is necessary. Accordingly, **associated sub-goals** of this PhD thesis deal with the identification of the necessary components to efficiently provide IDMS, with the analysis of the feasibility and suitability of different alternatives for such components, as well as with the design of novel components to enhance the global IDMS performance.

As a **first sub-goal**, this PhD thesis aims to analyze the emerging distributed media consumption paradigm, by exploring the relevant use cases and associated challenges. The intention is to identify the components and to derive the requirements that are needed to efficiently provide IDMS.

In relation with this sub-goal, associated Research Questions (RQs) are:

RQ1: Are delay differences in existing delivery technologies a barrier for realizing shared media experiences?

RQ2: Is IDMS a very specific research problem? Which distributed media applications would be benefited by the provisioning of IDMS?

RQ3: What are the requirements to efficiently provide IDMS?

As a **second sub-goal**, this PhD thesis aims to design the proper protocols to enable IDMS, while complying with the derived requirements. A key premise is to investigate if current standard protocols can be extended to provide IDMS. This would allow devising an evolutionary IDMS solution, being backwards compatible with existing technologies and systems.

In relation with this sub-goal, associated RQs are:

RQ4: Do any standard protocol fit the derived requirements for IDMS or can be extended to fit them?

RQ5: Is it feasible to independently and concurrently synchronize the media playout of several groups of users within the same shared media session?

As a **third sub-goal**, this PhD thesis aims to explore the feasibility and suitability of various architectural schemes for IDMS, in terms of key factors and deployment issues. Likewise, a related objective is the adoption of different centralized and distributed schemes, which will allow efficiently deploying our IDMS solution in a large variety of scenarios, according to the targeted requirements or available resources.

In relation with this sub-goal, associated RQ are:

RQ6: Which architectural schemes are best suited for IDMS?

RQ7: Can additional mechanisms be adopted to enhance the performance of existing architectural schemes for IDMS?

As a **fourth sub-goal**, this PhD thesis examines the feasibility and suitability of several dynamic strategies for choosing a reference timing to synchronize with. This selection may influence the overall quality of the media session, as it may have an impact on various key aspects, such as the synchronization effectiveness, interactivity, fairness, buffer fullness levels, frequency and magnitudes of the playout adjustments, etc.

In relation with this sub-goal, associated RQs are:

RQ8: Which strategies can be used for choosing the reference timing for IDMS (for each architectural scheme)?

RQ9: What is the impact of the application of such strategies?

As a **fifth sub-goal**, this PhD thesis aims to explore potential adjustment techniques to achieve IDMS. In particular, it analyzes if Adaptive Media Playout (AMP) techniques, typically used in other research areas, can be adopted for IDMS purposes. The aim is to devise a novel AMP technique to smoothly adjust the playout rate, within perceptually tolerable limits, every time an asynchrony situation needs to be corrected.

In relation with this sub-goal, associated RQs are:

RQ10: What are the benefits of adopting AMP for IDMS?

RQ11: Can the use of AMP avoid long-term playout discontinuities when performing IDMS?

Additional goals of this PhD thesis involve the selection of the most proper frameworks for implementing the components of the designed IDMS solutions, as well as for performing the required evaluation tests.

1.3 Methodology

In order to achieve the goals of this PhD thesis, the methodology sketched in Figure 1.1 has been followed. This figure shows the followed workflow, which has been divided into four main phases: 1) Problem Analysis; 2) Design Process; 3) Prototype Implementation; and 4) Evaluation.

The first step consisted of an exhaustive analysis of the distributed media consumption paradigm, by exploring the associated challenges, and the relevant use cases. Up to 20 use cases in which IDMS is necessary or beneficial were compiled and described. Likewise, a thorough study and classification of the existing IDMS

solutions was performed. This helped to understand the main components needed to accomplish IDMS, as well as the strengths and weaknesses of such solutions. This research phase aimed to show the relevance of IDMS, to reflect the need of this type of synchronization, and to determine the (technical) requirements that should be accomplished.

After deriving the key requirements that should be met, the design process of each of the components of the IDMS solution was initiated. This phase comprised the specification of the appropriate protocols, architectural schemes, control algorithms and adjustments techniques, as well as the design of various alternatives for each one of these components.

The IDMS solution has been implemented in prototypes in both a simulation and in real media framework. Such prototypes and the followed evaluation methodology are explained in further detail in Chapter 10. However, it is important to mention that, according to the schedule of the design process, the implementation and evaluation processes were not performed at a single stage, but repeated for each individual component of the IDMS solution under design, as well as for the global IDMS solution at a later phase.

A parallel phase thorough the entire duration of the PhD thesis consisted of the documentation and the dissemination of its research findings and contributions. Concretely, as a result of the intensive research work done, numerous papers in relevant national and international conferences, international journals, and workshops have been published. Likewise, posters describing the on-going work being done have been presented in doctoral symposiums, which helped to get valuable feedback about the research direction and methodology to follow. Moreover, some of the developed prototypes have been presented as demos in international conferences. Finally, the specification of the control protocol for IDMS has been standardized within the IETF, in RFC 7272.

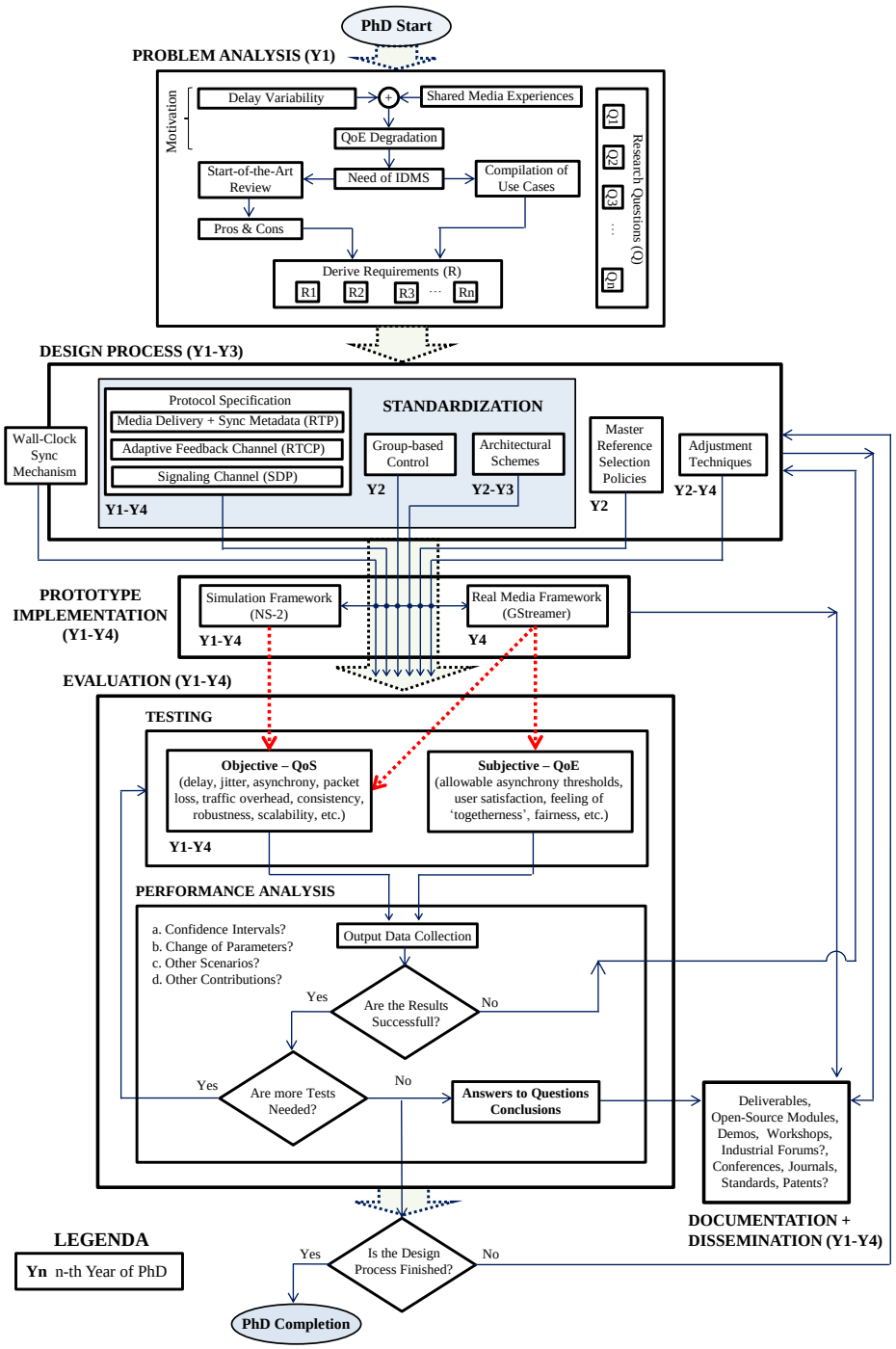


Figure 1.1. Workflow and Methodology of this PhD thesis

1.4 Structure of the Thesis

After presenting the context of this PhD thesis, the research goals and the employed methodology, this Section describes the structure of this memory, which has been divided into 12 Chapters and 1 Appendix.

Chapter 2 provides a detailed introduction to the multimedia synchronization research area, by defining key concepts and presenting the different types of synchronization, with relevant examples. Special attention is given to IDMS, as it is the particular temporal synchronization type this PhD thesis is focused on. Many use cases requiring IDMS are described to show its wide applicability and relevance in the current media consumption paradigm. After that, the main challenges to accomplish IDMS are discussed, by identifying various factors that can contribute to the lack of IDMS (and of multimedia synchronization, in general) when delivering media content over distributed scenarios. Moreover, it is shown that the magnitudes of the delay differences in actual delivery networks significantly exceed the allowable thresholds in the analyzed IDMS use cases. This corroborates the need of designing the appropriate technology to efficiently provide IDMS, which is the main goal of this PhD thesis.

Chapter 3 provides an introduction to the main components required to constitute a complete IDMS solution, and the required interactions between them. Likewise, the different options for such components are presented and their applicability is briefly discussed. Finally, an overview of assessment metrics for IDMS is also provided.

Chapter 4 provides a thorough review of the state-of-the-art in this research area. The first part of this Chapter includes an overview of different works that have surveyed existing multimedia synchronization solutions and proposed reference models to classify them. After that, the second part of this chapter is solely focused on IDMS, and provides a survey of the IDMS solutions that have been devised up to date, by classifying them in terms of key factors, according to many criteria and patterns from the surveyed reference models. Some of these classification factors were introduced in the previous chapter for a better understanding of this one.

In Chapter 5, the suitability and applicability of the identified architectural approaches and control schemes for IDMS is discussed. Likewise, a thorough qualitative comparison between the control schemes for IDMS is provided.

Chapter 6 presents a list of the key requirements for IDMS that have been derived as a result of the initial problem analysis phase (whose findings are in the previous chapters).

Chapter 7 presents an overview of the inherent capabilities of RTP/RTCP protocols to enable multimedia synchronization. Likewise, the relevance of such protocols in the current media delivery ecosystem is discussed. Finally, a summary

of the RTCP feedback reporting rules specified in different IETF standards is provided.

Chapter 8 presents all the different components (i.e., protocols, schemes, algorithms and adjustment techniques) of the RTP/RTCP-based IDMS solution designed in this PhD thesis. First of all, the rationale for using and extending RTP/RTCP for IDMS purposes is provided. After that, each one of the components of our IDMS solution, with their developed alternatives, their interaction with other standard mechanisms, and various particular operational aspects to enhance the IDMS performance, are presented.

In Chapter 9, the IDMS solution is extended by proposing a more strategic and efficient usage of the RTCP channel for IDMS. In particular, novel Early Event-Driven (EED) RTCP reporting rules and feedback messages, which in conjunction we call *EED RTCP Feedback for IDMS*, are specified with the goal of enhancing the performance of our IDMS solution in terms of interactivity, flexibility, dynamism and accuracy.

Chapter 10 presents the evaluation methodology that has been followed and the prototypes that have been implemented in this PhD thesis.

Chapter 11 includes the evaluation of all the components of the designed IDMS solution through simulation.

Finally, the general conclusions of this PhD thesis are included in Chapter 12. Moreover, it discusses some remaining and emerging challenges regarding the covered topics that need further research.

Likewise, Appendix A lists the publications that have been derived as a result of the intensive research carried out within the context of this PhD thesis.

To conclude, Table 1.1 provides a summary of each Chapter of this PhD thesis, and classifies the publications related with each one of them. In this Table, it is also indicated whether each contribution was published in Journal (J), Conference (C), Workshop (W), Book Chapter (B), Standard (S), or if it was presented as a Demo (D).

Table 1.1. Structure of this PhD thesis and derived publications

Chapter	Summary of Contributions	Publications
1	Context, research goals and methodology of the PhD thesis.	[Mon13c] (C)
2	Introduction to Multimedia Synchronization, Classification, Analysis of Challenges and Use Cases	[Mon12a] (J), [Bor12b] (J), [Bor13] (B)
3	Identification and Classification of IDMS Components. Overview of Adjustment Techniques and Quality Assessment Metrics	[Mon12b] (J), [Bor13] (B)
4	State of the Art: Overview of Reference Models, Compilation and Taxonomy of IDMS Solutions	[Mon12a] (J), [Mon12b] (J), [Bor13] (B), [Mon14c] (J)
5	Qualitative Comparison among Architectural Approaches for IDMS	[Mon12b] (J), [Bor13] (B)
6	Key requirements for IDMS	[Mon13d] (W), [Mon13c] (C), [Mon14c] (J)
7	Overview of the RTP/RTCP features for multimedia synchronization	[Mon10a] (C), [Bor11a] (J),
8	Design of the IDMS Solution, including all the components (protocols, schemes, algorithms, adjustment techniques…).	[Bor11b] (C), [Bor11c] (C), [Bor11d] (C), [Mon11a] (J), [Mon11b] (J), [Mon12a] (J), [Mon12b] (J), [Bor12b] (J), [Bor13] (B), [Bra14] (S), [Mon13a] (C), [Mon14c] (J)
9	EED RTCP Feedback for IDMS	[Mon13b] (C), [Mon15] (Internet draft)
10	Evaluation Methodology and Prototype Implementation	Simulation Framework: [Bor09b] (C), [Bor10] (C), [Bor11a] (J), [Mon10a] (C), [Mon10b] (C), Real Media Framework: [Mon14a] (D/C)
11	Evaluation Results	[Bor09b] (C), [Mon10a] (C), [Bor11b] (C), [Bor11c] (C), [Bor11d] (C), [Mon11a] (J), [Mon11b] (J), [Mon12a] (J), [Mon13a] (C), [Mon13b], (C), [Mon14c] (J)
12	Conclusions and Future Work	[Bor12a] (W), [Mon13d] (W), [Sto14] (Internet draft), and publications under review (see Appendix A).

Chapter 2

MULTIMEDIA SYNCHRONIZATION

2.1 Introduction

After presenting the context, research goals and methodology of this PhD thesis in Chapter 1, this Chapter introduces the multimedia synchronization research area, by providing definitions of key concepts and a classification of the different types of multimedia synchronization, with relevant examples for each one of them. From this classification, special attention is given to IDMS, as it is the specific type of temporal multimedia synchronization this PhD thesis is mainly focused on. A large number of use cases requiring IDMS is presented, with the intention to show its wide applicability and relevance in the current media consumption paradigm. After that, the main challenges to accomplish IDMS are presented, by identifying the different system components through the end-to-end media delivery chain that have an impact on delay and delay variability. Finally, it is shown that delay differences in actual delivery systems are significantly larger than allowable limits to enable coherent shared media experiences. This reflects the need of designing adaptive and accurate IDMS solutions to compensate for such delay variability.

2.2 Definitions

Multimedia systems are characterized by the computer-controlled integration of the generation, processing, communication and presentation of various types of media [Ste96]. The involved media types can be divided into two categories: continuous (also known as time-dependent and time-based) and static (also known as non-continuous, time-independent, non-time-based and discrete). Typically, the information of each media type is modeled as a sequence of Media Units or MUs (also known as Media Data Units or MDU, Logical Data Units or LDU, Information Units or IU, and Access Units or AUs), whose granularity is highly application-

dependent. Continuous media types, such as audio and video, are characterized by implicit and well-defined temporal relationships between subsequent MUs (audio samples and video frames, respectively), which are typically determined by the capturing/sampling processes. Such temporal relationships denote the order and duration of MUs. Static media types, such as text, images and graphics, have no implicit temporal properties. However, the temporal relationships between MUs of static media may have to be explicitly specified if integration with other media types is required. For instance, in a multimedia presentation, static media content (e.g., images or text) need to be presented at the correct point of time, just before, after, or simultaneously with, other static or continuous media, and during a specific period interval. Therefore, within this perspective, a multimedia system can be defined as "that system that supports the integrated processing of several media types, being at least one of them time-dependent" [Ste96]. Figure 2.1 illustrates a classification of multimedia systems, according to three criteria [Bla96]: number of involved media, types of media, and the degree of media integration.

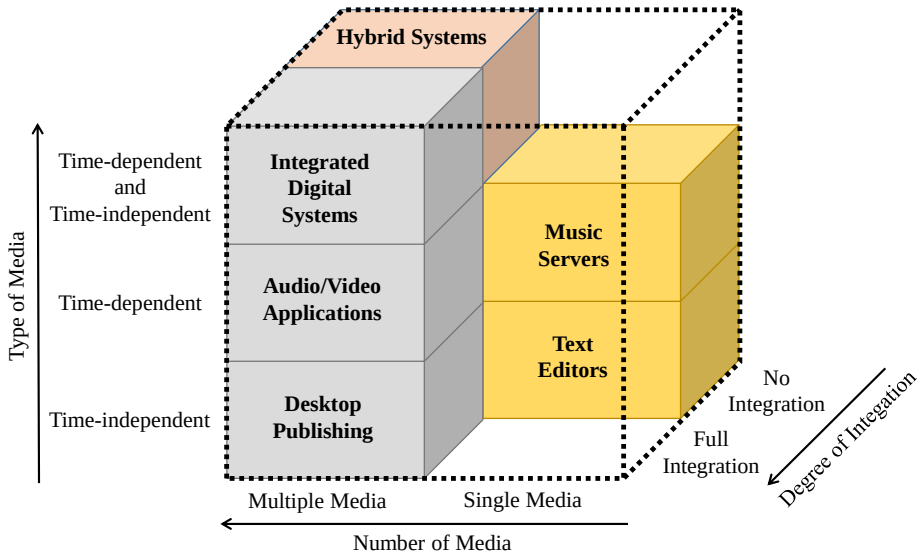


Figure 2.1. Classification of multimedia systems [Bla96].

Apart from temporal relationships, the involved media types (and subsequently their MUs) can have spatial and semantic relationships. On the one hand, spatial relationships refer to the physical arrangement of media data (e.g., layout of a multimedia presentation) as well as to the arrangement of the involved capturing and presentation devices (e.g., multi-channel audio, arrays of microphones and

loudspeakers, multiple TV cameras capturing different views of a scene…). On the other hand, semantic relationships refer to the content dependency between media types (e.g., two graphics with different interpretations or representations of the same data). Typically, media content with spatial and/or semantic relationships also have temporal dependences. For example, in a slide show, each slide can contain graphics, texts, animations, and even audio and video comments. In this case, the clarity of the presentation is highly influenced by the appropriate ordering, timing, layout and semantics of the multimedia information on the slides. If any of these dependences is not preserved, the slides can become difficult to understand, or can even convey incorrect information (e.g., audio comments referring to a graphic that has not still been shown or related graphics that are not shown with the correct order or position). Another relevant example is the simultaneous playout of different video streams containing different views of a scene (e.g., from different TV cameras in a stadium), as they have to be aligned in time and space.

Therefore, a precise mechanism of integration, coordination, and organization is needed in order to ensure, during presentation (or playout), the proper (implicit or explicit) dependences and evolution of the media types involved in a multimedia system. Such a process is referred to as *multimedia synchronization*.

Although multimedia synchronization encompasses the three above aspects (content, space and time), this PhD thesis focuses on the temporal dimension. Accordingly, within the scope of this PhD thesis, (temporal) media synchronization can be defined as "the process of guaranteeing the preservation of the temporal relationships between the MUs within and between the involved (continuous and/or static) media types in a multimedia system, based on their original timing attributes". Continuous media requires fine-grained synchronization, while discrete media allows more coarse-grained synchronization. Likewise, synchronization between continuous and discrete media is also relevant. For example, the audience of a slide show can be annoyed if a specific graph is shown before or after its audio explanation.

Synchronization is required when media is captured/retrieved and consumed within single devices, but it is especially relevant in distributed multimedia systems, in which media sources and receivers are not co-located. For instance, when streaming media over Internet, the MUs of the involved media types will be typically packetized and conveyed into single or multiple streams for transmission over the network. Such media streams will be sent (e.g., via unicast or multicast) by one or more sources to one or several receivers, which can (simultaneously) play out one or several of the involved media types. Each one of the packets of each media stream can follow different paths to reach the receiver/s and can be differently affected by (network and end-systems) jitter. Therefore, adaptive multimedia synchronization techniques are needed to reconstruct the original timing for each of the incoming media streams at the receiver side, typically with the help of playout buffering strategies. To achieve this, specification of such original temporal relationships at

the source side is necessary (e.g., by inserting specific metadata into the MUs or delivery packets). Accordingly, collaboration between the involved entities in the multimedia distribution chain (e.g., sources, receivers, and even inter-network devices) is required to efficiently accomplish multimedia synchronization. Likewise, multimedia synchronization must be addressed and supported by many system components, including hardware devices, Operating System (OS), protocol layers, storage systems, multimedia files, and even by applications. Hence, synchronization is an end-to-end challenge (i.e., from capturing to consumption) which must be addressed at several levels in a multimedia system.

2.3 Classification of Temporal Multimedia Synchronization Types

Based on the real-time nature of the media content and on the specification of the temporal relationships between MUs, two types of multimedia synchronization can be distinguished: live and synthetic synchronization. In live synchronization, media content is "live" captured from a real-time sensor (e.g., camera, microphone…). In synthetic synchronization, media content is retrieved from storage systems, although probably it was originally "live" captured and then stored. In the former case, the capturing and playout processes are performed during a continuous temporal process (with a delay as short as possible); in the latter case, MUs are captured, stored and played out at a later point of time. Live synchronization attempts to exactly reproduce during playout the temporal relationships between MUs that were specified during the capturing process. However, such original temporal relationships can be altered in synthetic synchronization, according to the available resources or targeted requirements (e.g., by changing the transmission or playout rate). Synchronization is easier to achieve in synthetic synchronization than in live synchronization, because of the softer, or even inexistent, real-time requirements and higher flexibility (e.g., it is possible to adjust the transmission and playout rates, schedule the initial playout times as desired…). Conferencing is an example of a live synchronization use case, while synthetic synchronization is often used in retrieval-based services, such as Content on Demand (CoD).

In both live and synthetic synchronization, three main types of temporal multimedia synchronization techniques can be distinguished, namely: intra-media (also known as intra-stream and serial) synchronization, inter-media (also known as inter-stream and parallel) synchronization, and Inter-Destination Multimedia Synchronization or IDMS (also known as group, multi-point, inter-participant and inter-receiver synchronization). Such classification is based on the number of involved media types, sources, streams and receivers. An example of each multimedia synchronization type is shown in Figure 2.2, in which a group of distributed receivers over an IP network are playing video, data (e.g., text chat) and audio streams.

First, *intra-media synchronization* deals with the maintenance, during the playout, of the temporal relationships among subsequent MUs within each media type. In Figure 2.2, we can observe a proper and continuous playout process of each media stream in all the receivers, such as the evolution of a video sequence showing a jumping ball, together with the data and audio streams. The goal of intra-media synchronization is that the temporal relationships between MUs during presentation (at the receiver side) resemble as much as possible to the ones specified by the capturing/sampling and/or encoding processes (at the source side). As an example, if the media source captures a video sequence at 25 MUs (video frames) per second, each MU must be played out (displayed) during 40 ms at the receiver side, as shown in the figure.

To achieve this kind of synchronization in distributed multimedia systems, playout buffering strategies at the receiver side are typically employed. On the one hand, the size of the playout buffer must be large enough to compensate for the effect of network jitter. On the other hand, the buffering delays need to be as short as possible in order to minimize the latency of the multimedia service. Likewise, the playout buffer occupancy must be kept into stable and safe ranges, such that buffer overflow and underflow situations are minimized. This way, a continuous and smooth playout for each media type can be guaranteed. Otherwise, the lack of intra-media synchronization can cause annoying playout discontinuities or interruptions (e.g., image jerkiness or audio distortion).

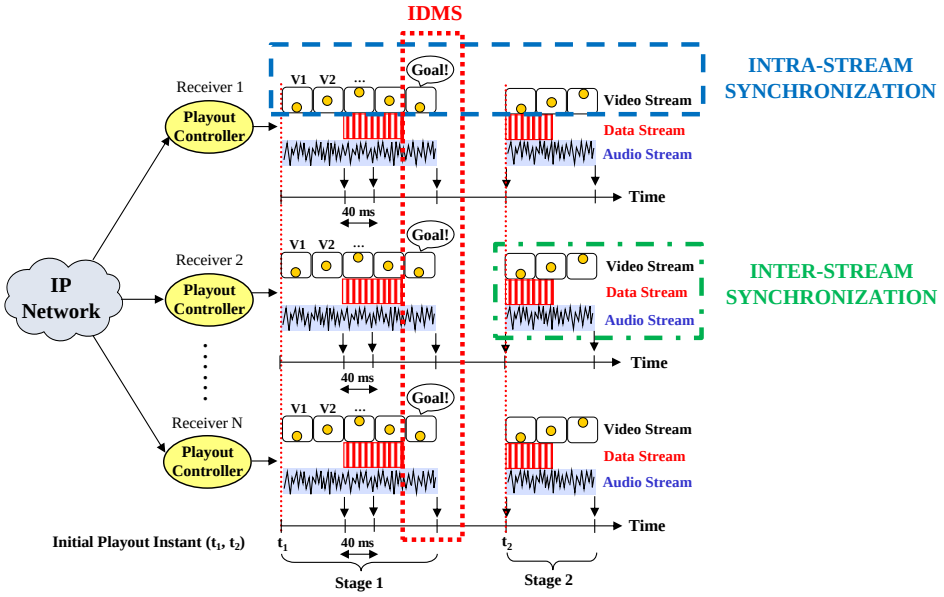


Figure 2.2. Examples of Multimedia Synchronization Types.

When several correlated media types are involved in a multimedia system, the original temporal dependences between their MUs must also be preserved during playout. That is the goal of *inter-media synchronization*. This is not an easy task, since the different media types may have different quality, processing, storage, communication, and presentation requirements. Obviously, in order to perform inter-media synchronization, each one of the media types must be also individually synchronized by means of intra-media synchronization.

Several use cases of inter-media synchronization can be cited. The most relevant one is audio/video synchronization. In such a case, video frames and audio samples captured at the same time must be also simultaneously presented at the corresponding output devices (ideally with the minimum latency in time-sensitive services). For example, in remote user's speech, the synchronization between the user's audible words and the associated movement of his or her lips is necessary. This is commonly known as *lip synchronization or lip-sync* (e.g., [Che03] and [Bar07]). A similar inter-media synchronization scenario is when using a two-lens stereo camera system with an internal mono microphone. In such a case, if only 2D video streaming is required, the synchronization between the audio and the video content from one of the lens will be sufficient. However, if 3D video streaming is required, synchronization between the video content from the two lens will also be necessary. A third use case is the synchronization of subtitles with the appropriate audio-visual content (e.g., [Rod12], [Con13]). This is useful for karaoke systems, for users with hearing impairments, or to provide the translation of the audio content to a desired language. A fourth use case is the synchronization of audio-video content with computer-generated scent (i.e., olfactory data), which is gaining relevance for enabling immersive multimedia experiences (e.g., [Ade09], [Mur13], [Mur14]). The last example being cited is the synchronization between audio/visual content and haptic media in networked games and in 3D virtual environments (e.g., [Hua14]). In each of the cited examples, the lack of inter-media synchronization can originate confusing and inconsistent media presentations (e.g., audio comments related to an incorrect video scene or graphic).

In [Hua13], the term intra-media synchronization is also used to refer to the synchronization between different media types of the same modality, which is a specific use case of inter-media synchronization. This is required when arrays of capturing (e.g., cameras or microphones) and output devices (e.g., screens or loudspeakers) are involved in distributed scenarios. Such arrays of sensors and output devices typically capture and present, respectively, media content from different positions or angles, with the goal of providing enriched and immersive media experiences.

Two main approaches can be followed when several media types are sent by the same source. The first one is to multiplex the packetized MUs of each media type into an aggregated stream (see Figure 2.3.a.1), whilst the second one consists of sending an individual stream for each involved media type (see Figure 2.3.a.2).

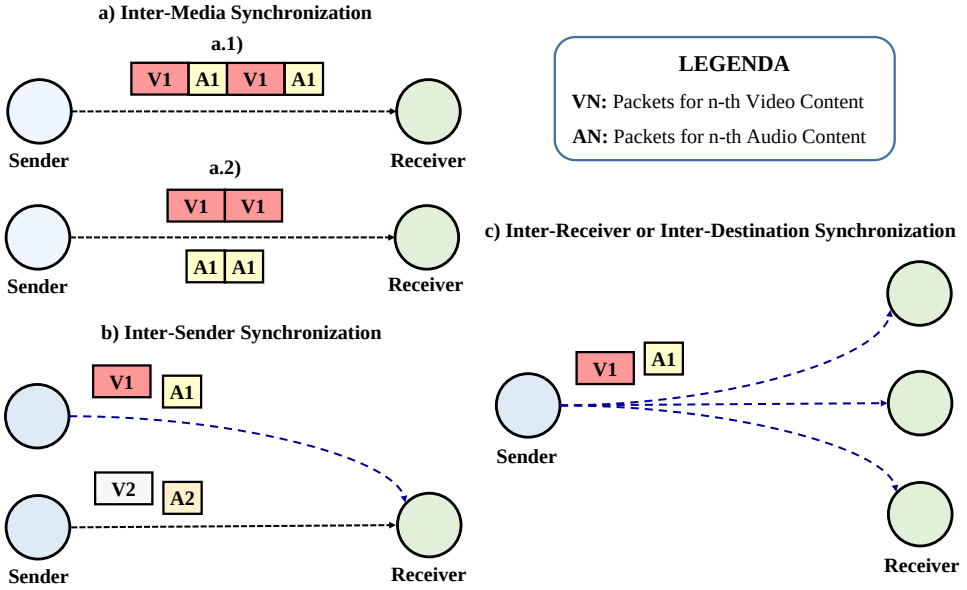


Figure 2.3. Multimedia Synchronization Scenarios.

A specific sub-type of inter-media synchronization is referred to as *inter-sender* (also known as multi-source) *synchronization* (see Figure 2.3.b), which aims to synchronize the playout of several media types originated from different senders (or sources). A typical example is the synchronization between two video streams sent by different video servers (e.g., in surveillance systems).

Moreover, it can also be possible that the different media types, sent either by the same or different senders, are delivered using different protocols, or even via different (e.g., broadcast and/or broadband) networks. This particular sub-type of inter-media synchronization is typically referred to as *hybrid synchronization* (e.g., [Con12], [Bel12c], [Vee12]). Two representative examples can be cited. The first one is the simultaneous playout of the video streams of a sports event being transmitted by broadcast (e.g., via Digital Video Broadcasting or DVB) and broadband TV operators (e.g., via Internet Protocol Television or IPTV). The second one is the synchronization of real-time statistics or subtitles from a web server with audiovisual content from a TV provider.

In specific cases, the involved media types can be played out on separate devices (e.g., multi-screen applications), so a new sub-type of inter-media synchronization can be distinguished. This is usually known as *inter-device synchronization* or IDES (e.g., [Bra13]).

The third type of multimedia synchronization is IDMS, which refers to the simultaneous synchronization of one or more playout processes of one or several media types across separated destinations or receivers (see Figure 2.3.c). For example, it can be noticed in Figure 2.2 that at any moment during the multimedia session all the receivers are playing the same MU of each media type. IDMS can be applied to any type and/or combination of media types, such as audio, video and scene information (e.g., chat, subtitles, images...). This type of multimedia synchronization has recently gained relevance to enable coherent shared media experiences, such as multi-party conferencing, Social TV and networked Multi-player Online Games (MOGs). In the next Section, a large number of use cases in which IDMS is needed will be presented.

In IDMS, the involved receivers can either be physically close-by or far apart. An example of the former is when the receivers are placed within the same local environment (e.g., different TVs in a home or loudspeakers at an airport)[1], whilst an example of the latter is when (various groups of) geographically distributed friends are watching an online sports event, while chatting (e.g., via an audio, video or text channel). In the latter scenario, the delay differences from the media source to each one of the involved receivers will be typically higher than in the former one. Likewise, in both scenarios, the involved consumption devices may have different bandwidth and/or processing capabilities. This issue will also significantly contribute to increase the end-to-end delay differences between them.

When multiple media types are involved in a distributed multimedia systems and IDMS is pursued, the general approach is to only perform IDMS on the stream carrying out a specific media type (which is known as primary or master stream), and then perform local inter-media synchronization mechanisms for the other streams (which are known as secondary or slave streams), according to the playout timing of the master stream. The master stream can be, for example, the one carrying out the audio or the base layer when using multi-description (or layered) media. The research in this PhD thesis focuses on IDMS for a shared stream (i.e., when multiple receivers are consuming the same media sent from a single sender), as shown in Figure 2.4. However, it could also be possible that the different receivers need to be synchronized together (IDMS), but are not consuming different media. For instance, they can be consuming the same media content in different formats (e.g., using different encodings or the same encoding with different settings), video content captured by different cameras (e.g., in different positions of a stadium or from different TV operators), or even different media types (e.g., audio and video). These particular IDMS use cases are not covered in this PhD thesis.

[1] The term IDES is sometimes also used to refer to this multimedia synchronization use case, but as a subset of IDMS rather than as a subset of inter-media synchronization.

As a summary, Figure 2.4 shows a taxonomy of the different types of multimedia synchronization, according the involved media types, senders and receivers.

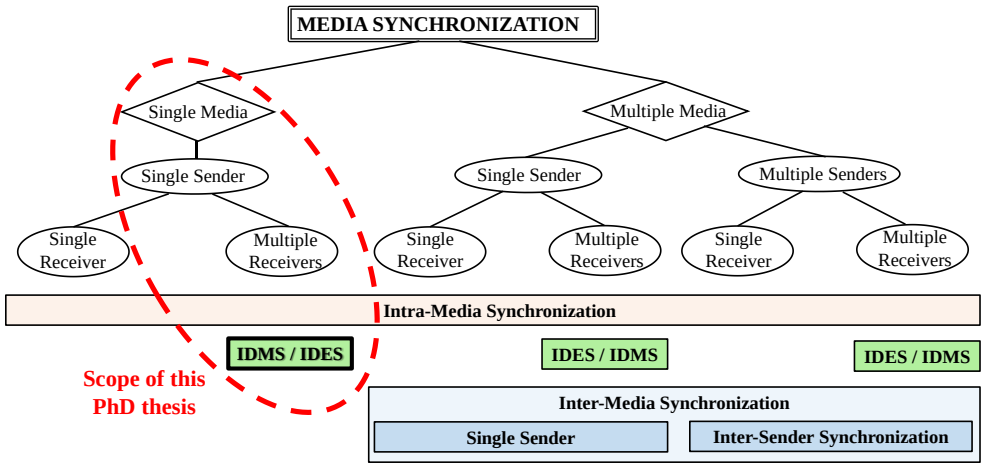


Figure 2.4. Taxonomy of Multimedia Synchronization: Focus on IDMS.

2.4 IDMS Use Cases

Nowadays, we can find many distributed multimedia applications in which the lack of IDMS may affect the QoE in different ways. In this Section, a large compilation of IDMS use cases is presented to show its wide applicability and relevance:

1. *Synchronous e-learning*. In real-time distance learning applications (e.g., the one proposed in [Pre10]), an instructor can distribute a multimedia lesson to a group of students that could be attending it from different locations. In such scenario, the instructor can occasionally make some comments or questions about the content of the lesson. Hence, it is very important that each one of the students receives such questions (possibly transmitted in several media streams) at the same time and, as a result, has a fair chance of answering.

2. *Networked quiz shows* with distributed on-line participants, in which the winner is the first one to answer a multimedia question correctly. In such a case, in absence of IDMS, participants may feel unfairness because the contestant with the shortest delay will have an advantage over the others. National laws may even prohibit this, as broadcasters are not allowed to offer games of chance without a specific license for this, and without IDMS such quiz shows may become a game of chance.

3. *Networked real-time multiplayer games* (e.g., [Dio99], [Mau04], [Roc08], [Hos09]), where multiple media types, such as computer data (e.g.,

information input from a keyboard or a haptic device), audio and video are simultaneously involved. In such scenarios, multiple players often collaborate (as a team) with each other and/or fight against other multiple players (belonging to other teams). If a player presents output timing different from the other players, the fairness among them, or the efficiency of the collaborative work, can be seriously damaged.

4. *Multimedia Cluster-to-Cluster*[2] (*C-to-C*) applications or multi-point to multi-point communications (e.g., [Ott07], [Bor09c]), including independent but semantically related media streams (e.g., audio, video, text, sensory data…) sent from end-systems located in one or more clusters (sender clusters) to end-systems located in other distributed clusters (receiver clusters). For example, the sender clusters may consist of a collection of capturing devices (e.g., video cameras, microphones...), each one producing an independent stream of data, and the receiver clusters might be a collection of computers that process and store the incoming data streams as well as consumption devices (e.g., screens, speakers...). Likewise, bidirectional communications, in which each cluster has both transmission and reception capabilities, are typically supported. Examples of such applications are: computer-supported collaborative environments [Kim05], video-centered communications (e.g., surveillance systems, traffic and street monitoring...), 3D Tele-Immersion (3DTI) [Hua11, Hua12], etc. For instance, in a 3DTI scenario, a scene acquisition sub-system could be comprised of an array of digital cameras and computing hosts set up to capture a remote physical scene from a wide variety of angles. All the captured videos would be concurrently multi-streamed to a distributed 3D reconstruction sub-system at a remote location. The resulting view-independent depth streams would be used to render a view-dependent scene on a stereoscopic display in real-time using head-tracking information from the user. Overall, the application would allow remote participants to interact within a shared 3D space, so everyone would feel a strong mutual sense of presence.

   All these C-to-C applications pose sophisticated data transport requirements due to the use of multiple, semantically related, media streams. Therefore, synchronization mechanisms (including IDMS) must be provided to guarantee a high quality multimedia system, regardless of the number of receivers and streams consumed on the receiver clusters.

5. *Distributed tele-orchestra* (e.g., [Sch93], [Miy11]). IDMS can enable the simultaneous playout of a music orchestra at different locations, by

[2] A cluster can be considered as a collection of computing and communication end-systems sharing either the same local environment or a media experience as a logical group.

remotely synchronizing all the correlated media streams from multiple live musicians located in various geographically distributed sites. The orchestra may consist of as few as a couple or a trio of live musicians (e.g., the scenario in [Sch93]) to an entire orchestra with many musicians. As a conductor (reference), one (preferably continuous) pre-recorded media stream or a metronome stream could be used, thus providing an aural cue. That reference media stream (e.g., a piano symphony) may be originated from one site and sent to the other sites where live performers are listening to it and playing their corresponding instrument melodies in a temporally synchronized way, which will be transmitted in new individual media streams. Additionally, if needed, the reference or metronome stream could also be forwarded by one of the remote sites. Note that neither the performers nor the conductor could hear the compound symphony entirely. Each performer could only hear the conductor part of the orchestra (a somewhat contrived musical experience for the performers). The correlated media streams must be delivered to the audience in a synchronized manner to provide a high-quality music performance in spite of delay variations and network fluctuations. Moreover, those media streams must be also played out simultaneously at all the distributed listeners' locations. This scenario imposes very stringent synchronization requirements to achieve a high-quality music orchestra, compounded by the individual melodies from distributed live musicians.

As a similar use case, the effect of IDMS control in a networked chorus was studied in [Miy11]. In this scenario, there was a conductor providing a standard timing, several geographically distributed singers singing according to that standard timing, actions of the conductor providing instructions to the singers, and a group of distributed listeners as an audience. In such scenario, IDMS is needed in order to coherently present the actions of the conductor and the overall singing voices in each one of the singers' and listeners' terminals, respectively. The assessments results proved that IDMS can significantly improve the overall user experience (QoE) in a networked chorus.

6. *Multi-party multimedia conferencing*. In these applications, if the output timing of audio and video by a participant largely varies from destination to destination, the conference itself cannot be held. Furthermore, if the number of participants becomes large, the playout delay differences may increase.

7. *Consumer-originated content and content sharing on a multimedia conference*, whose purpose is sharing content in real-time with family, friends, colleagues or other types of "buddies" all over the world. An example is when browsing together through recorded digital photos and videos and commenting on the content in real-time.

8. *Conferencing sound reinforcement systems*, often used in commercial and government installations, such as legislative chambers, courtrooms, boardrooms, classrooms (specially, those supporting distance learning), etc. Each participant who is using one of these systems has a microphone and a speaker. There may also be other speakers to provide reinforcement for non-speaking participants, such as in an audience area or jury box. Each microphone/speaker pair is individually connected to a network, transmits digital audio over the network to the other devices, and receives digital audio to be reproduced through the speakers.

   Likewise, there may be a central appliance which receives, prioritizes, and mixes the microphone signals. In some systems, an individual mix is created for each speaker so the speaker's own voice does not come out from his/her loudspeaker or from those immediately surrounding him/her. The objective of these systems is not that the person speaking sounds or feels amplified so much as it is to provide enough gain to enhance intelligibility. Reaching this objective helps to ensure that natural person-to-person communication is retained. To this end, it is desirable that the sound through the system and from the speakers arrives 5-30 ms after the sound arriving through the air from the person speaking. Delays in this range invoke the Haas effect [Haas Effect] which allows listeners to locate the person speaking based on the sound arriving through the air, while the sound reinforcement system provides the additional gain required to achieve the desired intelligibility. It is also desirable for the sound to come out of nearby speakers at within 5 ms as longer differential delays will be perceived as reverberation or echo.

9. *Networked stereo loudspeakers* in which two or more speakers are connected to the network individually. Humans can localize sound based on inter-aural time differences in a stereo listening situation. Therefore, humans are very sensitive to changes in latency between the (two) speakers. We perceive these changes as a shift in or instability of the "*sound stage*" during critical listening. Shifts around 10 µs (or even smaller) could be noticeable. If the individual speakers operate from independent network interfaces in a stereo listening setup, any changing difference in latency between the (two) speakers greater than few microseconds will negatively affect the listening experience.

10. *Phased array transducers* used in audio applications. This technique works by sending or receiving slightly different versions of a signal in a spatial sampling arrangement to produce or record spatial and directional sound fields. One example application is the conferencing microphone system that is able to electronically aim at the person speaking to improve signal to noise ratio. These microphones are also able to report the location of the speaker for purposes of automatically aiming a video camera at them. The

individual transducers in such applications can be extremely sensitive to differential latency. Another example is a concert sound system called "line arrays" which allows technicians the control over the amount of sound sent to different places. People in front of the audience can have the same loudness as those in the back. By preventing sound from reaching the roof and back wall of the performance space, the amount of reflected sound heard by the audience is reduced and the listening experience is improved. In these systems, accuracy in locating or emitting sound is related to differential latency through basic trigonometry. Microseconds of differential latency can translate to degrees of uncertainty. Accuracy greater than the audio sample period (about 20 μs for professional 48 KHz sample rate) is generally desired.

11. One of the most prominent use cases in which IDMS becomes indispensable is *Social TV*. This enables different groups of viewers, independently of their location, the network and the device they are using, to watch a TV program, while simultaneously interacting and sharing services, via chat messaging, audio/video conferencing services, or for that matter any other sort of shared experience that is yet to appear. In [Ces09a] and [Vai11a], various media streaming applications providing some form of synchronous shared experiences are presented. As an example, Watchitoo[3] is a web-based application that enables not only chatting, but also audio and video conferencing while watching the same TV content.

    What started as Internet TV has evolved into a richer mix of media for Social TV, allowing direct social interaction among people, supported by two-way communications. Social TV combining TV content with direct social and community interaction (e.g. using Facebook, Twitter…) is taking root in connected Set Top Boxes (STBs), web-ready TVs, and PCs. The traditional ubiquitous model (two children and mom-and-dad scenario), obsolete and overused, is being replaced by a much more dynamic family unit that is spread around the world with people moving and interacting digitally. TV is part of the shared family experience and will continue as a part of its heritage. As people are social by nature, this new TV model promises to deliver a world of content and services to any combination of devices, anywhere and anytime (the future of IPTV is connected, mobile, personal and social [Mtp11]).

    A typical scenario is when various friends are watching a live on-line football match at separate locations (*"watching apart together"*), as reflected in Figure 2.5. We could also think about the possibility of adding more friends to the shared session, for example, those who are traveling by

[3] http://watchitoo.com/

train, viewing the match using smartphones and, in an extreme case, some other friends could be watching the match live physically at the stadium and communicating with the others using their smartphones (audio/video calls or text messages). In such a case, inter-media synchronization must be performed between the involved media streams, such as between the TV content the users are watching together (e.g., the video and audio corresponding to a football match) and the associated communication streams between the users (e.g., audio/video conferencing). Moreover, significant events, e.g. a goal (see right side of Figure 2.5), should be perceived by all the users almost simultaneously (IDMS), even in all the associated interaction streams, to not degrade the user experience on such interaction. Instead, it would be very frustrating for a home user to experience a goal later than the friends at their homes (or train) while they are chatting.

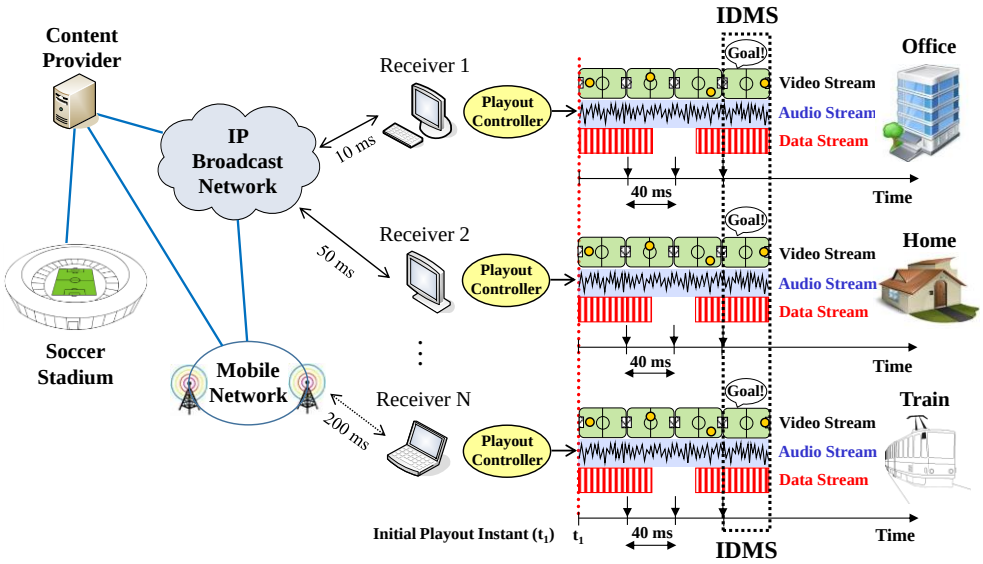


Figure 2.5. A Generic Social TV Use Case.

Similarly, the work in [Hes10] revolves around a socially augmented rock concert in which various friends share the music experience and enrich it through social interaction and media sharing. Some of the friends are watching a live broadcast of the concert (high-quality professional TV content), each from their own home. They could talk to each other using the IP-based communications facilities built into their TV sets (via Internet) and at the same time receive a live video feed from some other friends actually

attending the concert. The friends at the concert would use their smartphones to generate the video stream, which could be rendered as a picture-in-picture overlay on the TVs of the remote friends, giving them a view of the concert from the local audience's point of view. Besides, the friends can interact with each other and comment on the shared music experience via text chat or audio/video conferencing.

To enable this kind of services, some platform, such as the one presented in [Hes10], will be needed. This platform will be used for creating a dynamic community involving all the users sharing the media experience. A cross-domain session will be established, through which media and interactions will be shared, synchronized, adapted, recorded, played back, and analyzed (with the consent of the users). This session would exist for the duration of the shared session and any related activities (e.g., post-match advertising in the shared football experience). Once the group has been created, all the users should be informed in an appropriate way, based on their context. Likewise, once the shared session begins, the users could talk to each other and discuss about it, including watching each other (video conferencing). In the shared football experience, friends at the stadium could send videos of the match to give friends at home a view of the match from the spectators' point of view, whereas friends at home could also share the recorded TV edited highlights (e.g., to clarify off-side situations).

12. *On-line election events*. As an example, in a pop star competition show, any vote from viewers (fans) at home sent during the show must be valid, and all the votes sent after the deadline (lines are closed) must be rejected.
13. *Presence based games*. In such scenarios, users can win a prize when they watch a certain advertisement at a certain time. When the content is too much out of synchronization, it can no longer be determined what specific content the user has been exactly watching.
14. *Game-show participation*. Starting from simple messaging to a TV show or dialing in by phone, users will become live participants in TV shows with live streaming footage through user webcams and real-time interaction between the participants and the TV show.
15. *Shared service control*. This use case is similar to Social TV and allows distributed users to experience CoD together, while sharing the trick-play controls (play, pause, fast forward, and rewind). Differences in playout speed and the effect of different transit delays of MUs and of trick-play control signals would de-synchronize content playout.
16. *Multi-Screen Settings.* TV viewing is also becoming a multi-screen proposition. Nowadays, it is quite common that in the same living room, apart from the (shared) main TV or big screen, each family member is also

watching the TV content on their personal device (e.g., tablet, smartphone or laptop). Besides, single users often use secondary devices to consume extra, but related, content while watching TV. A typical example is when different screens are playing out video streams from different cameras in a stadium or in a circuit race. Other relevant use cases of multi-screen settings are community gaming around TV content, rating systems for talent shows and interactive quiz shows.

In such multi-screen experiences, users can directly perceive delay differences. Hence, the synchronization requirements become stricter than when the content is consumed by geographically distributed users.

17. *Seamless switching among media devices*. When users switch their multimedia session between different devices (e.g., from a fixed TV set to a mobile device), a smooth and seamless transition must be provided. If there is too much delay difference between the presentation times in the involved devices, this will spoil the switching experience, as a significant portion of the content may be missed or played out twice.
18. *Networked video walls*. A video wall consists of multiple computer displays, video projectors, or TV sets tiled together contiguously or overlapped in order to form one large screen. Each screen only shows a part of the larger picture. In some implementations, each screen may be individually connected to the network and receive its portion of the overall image from a network-connected video server or video scaler. Screens are refreshed at 50 Hz (i.e., every 20 ms) or potentially faster, but if the refresh is not synchronized, the effect of multiple screens acting as one will be broken.
19. *Synchronous groupware*. This is a technology that facilitates teamwork, supporting the communication and coordination between geographically distributed team members [Luk03]. It encompasses a wide range of applications like collaborative whiteboards or text editors. These applications need to share a consistent common state to enable an efficient integrated collaboration environment.

2.5 Challenges: Delay and Delay Variability Factors

Multimedia content distribution platforms are known to introduce delay. This is specially an issue in digital communications. Moreover, present-day IP-based networks provide no guarantees, either on delay or on delay variability (i.e., jitter) bounds. Delay is not a serious constraint when isolated users are consuming non-time-sensitive content (from either broadcast or broadband networks). Nevertheless, delay, jitter, and delay differences, both between streams and receivers, become serious barriers when tight real-time requirements must be met, and when

interactivity between the users and the media content, as well as between users (within the context of specific media content consumption), are pursued.

Various system components belonging to different steps of the multimedia delivery chain contribute to the end-to-end delay and delay variability, thus having a significant impact on synchronization. Such sources of delay and of delay variability can be originated at the sender, distribution, and receiver sides, and mainly depend on: i) the features of the involved devices in the delivery chain (sources, receivers and inter-network equipment); ii) the media types and encoding settings; iii) the network infrastructure through which media is delivered; iv) the network and CPU (Central Processing Unit) load of the involved devices; and v) clock imperfections.

Figure 2.6 shows the impact of various system components on the end-to-end delay and on the delay differences when delivering media. Most of such factors introduce variable and undesired delays, and their behavior is very difficult to predict and control. However, a proper understanding of the origin of these sources of delay and of delay differences, as well their integrated treatment, is essential to efficiently devise multimedia synchronization solutions.

On the one hand, it is shown that end-to-end delay differences when streaming different media types (from either the same or different sources) to a single receiver lead to the need of inter-media synchronization. Without inter-media synchronization, MUs of different media types generated at the same instant will not be simultaneously played out at a specific receiver. On the other hand, it is shown that end-to-end delay differences when delivering the same media content to distributed receivers lead to the need of IDMS. Without IDMS, the same MUs of specific media types will not be simultaneously played out at the different receivers. If the playout time differences, both between media types and between receivers, exceed allowable thresholds, then interactive multimedia services may be impossible to be provided.

Next, the most relevant sources of delay and delay differences in each stage of the multimedia distribution chain are briefly described.

Delay at the sender side: The computation heterogeneity of the different servers, as well as the temporal CPU load variation in each one of them, can cause variable delays at the sender side. At this stage, various sources of delay and delay variability can be cited, such as: capturing, sampling, encoding, encryption, packetization, protocol layer processing and transmission buffering. For instance, the capturing of different types of media can take different amount of time, as different computational resources are required for each one of them (e.g., audio and 3D video), mainly due to the amount of captured data and to the (settings of the) encoding mechanisms being used. Likewise, the use of different delivery technologies can incur in different transmission overhead. Measurements of delays and delay variability at the sender side can be found in [Hua13].

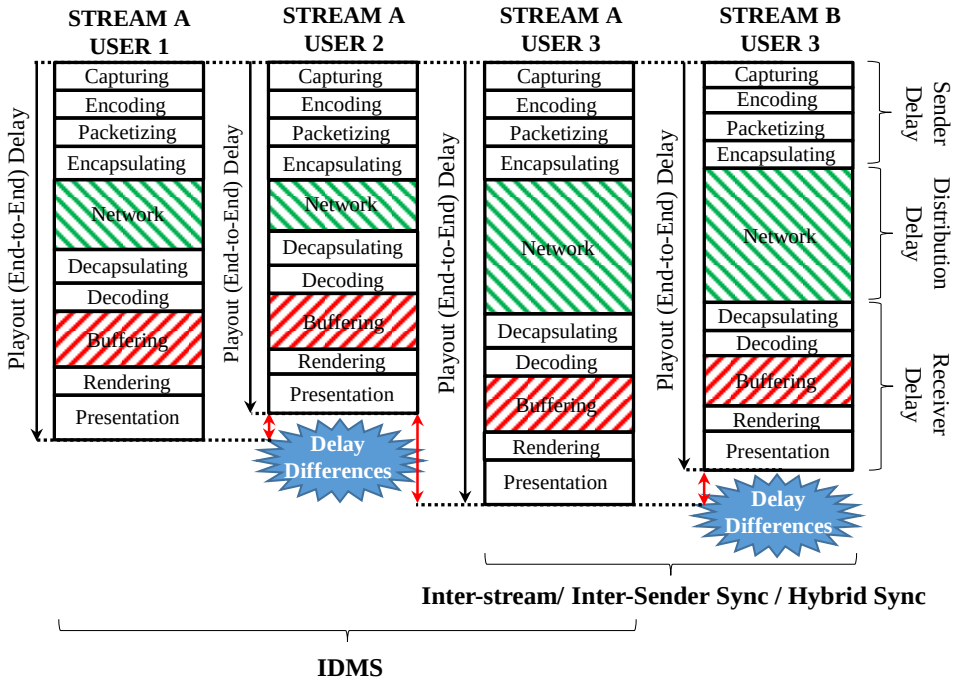


Figure 2.6. End-to-End Delay Variability: Need for Multimedia Synchronization.

Network delay: It is the delay experienced by the MUs of a specific media stream sent from a source to reach a specific receiver, which varies according to the network load. The network delay includes the propagation and serialization delays through the involved links, as well as the processing (e.g., routing decisions, queuing policies…) at the intermediate routers. Likewise, advanced procedures, such as fragmentation and re-assembly of packets, trans-coding and format conversion, can also occur at this stage and have an impact on the delay and delay variability. Network jitter denotes the variation of inter-arrival times of MUs at the receiver side. It is mainly due to the variation of the network load and of the connection properties (this is especially relevant in wireless networks). For example, as a result of an increase of the network load, the links and the intermediate routers can become congested, with a consequent increase of the propagation and processing delays, respectively. Even, data packets may be lost during distribution and may not arrive to the targeted receiver/s. Network jitter may destroy the original temporal relationships between MUs of each individual stream and of different related streams. Typically, an elastic playout (also known as reception and de-jitter) buffer is allocated at the receiver side to compensate for the effect of network jitter.

When a single sender transmits various media types in an aggregated stream, e.g. when using MPEG2-TS (Moving Pictures Experts Group 2 Transport Stream), then the network delay differences between media types are non-existent. However, if the different media types are sent as individual streams, e.g. when using RTP/RTCP protocols [Sch03], the packets of each stream may follow different paths to reach even the same receiver. Moreover, if different delivery technologies (e.g., broadband and broadcast, or different broadband protocols) are used for each media type, the delay differences will be typically higher.

Delay at the receiver side: Delay and delay variability at the receiver side are mainly originated due to buffering, depacketization, protocol layer processing, decoding, decryption, and rendering processes. Likewise, any type of error and/or loss concealment techniques will incur in additional delays. The buffering and decoding delays are the most relevant sources of delay at the receiver side. For example, the decoding delays can be significant in those encoding mechanisms using interpolation methods and large Group of Pictures (GOP) sizes. The total delay at the receiver side can also fluctuate over time, according to the instantaneous CPU load of each receiver, to the real-time responsiveness of the OS, to the protocol layer processing, as well as to the current state of the network (as it determines the rate at which data packets are received). This variable delay is usually known as *end-system jitter*. Moreover, the delays at different receivers can be different, mainly due to their heterogeneous software and hardware resources and to possible different settings (e.g., different buffering delays in each receiver).

An additional factor to take into account when using digital TVs is the display lag, which is the time difference between the instant at which a video signal is input into a display and the instant at which it is shown by the visualization device. It may be originated by image processing routines, such as scaling and enhancement. The display lag may also cause a noticeable offset (i.e., delay differences or asynchrony) between the audio and the image signals, thus having an impact on inter-media synchronization. Display lags in High Definition (HD) TVs can vary between 30 and 90 ms, depending on the TV model and on the type of input signal [Mek11]. In this context, the study in [Jan13a] analyzed the impact of different TV modes (concretely, "Dynamic" and "Game" modes) on the video delay and delay variability. For the TV being tested, it was shown that delays when using "Game" mode were almost 90 ms shorter and quite a bit less variable (at the cost of a slight degradation of the video quality, according to the TV documentation, although it was unnoticeable). Concretely, in the scenario under test, when enabling "Game" mode, the average end-to-end video delay was 357 ms, and the standard deviation was 20 ms, while when enabling "Dynamic" mode the average end-to-end delay was 442 ms, and the standard deviation was 36 ms. Accordingly, the features and settings of the specific consumption devices have a significant impact on the delay and delay variability.

Clock imperfections: The availability of precise, high resolution and reliable timing mechanisms is a key aspect in distributed multimedia systems. Media capture, encoding, transmission, decoding, presentation, and many other processes are driven by end-system clocks. Likewise, if the clocks of the involved senders and receivers are not in perfect agreement, or do not run at the same rate, multimedia synchronization issues can arise. Regarding intra-media synchronization, buffer overflow (flooding) or buffer underflow (starvation) situations can occur if the receiver's clock is slower or faster than the sender's clock, respectively. Regarding inter-media synchronization, asynchrony situations will arise if the clock rates of the involved senders and receivers do not match (e.g., in inter-sender synchronization or in IDES). Regarding IDMS, asynchrony situations will occur if the receivers' clock are not time-aligned. Within this context, the *clock offset* (also known as skew) refers to the time differences between the clock instances of two entities, while the *clock drift* is the rate of change of the clock offset, due to a non-homogeneous advance of the clock rates (i.e., the clock frequency varies over time). This fluctuation is very close related to the resolution of the crystal clocks, oscillator stability, voltage changes, aging, surrounding temperature and other environmental variables (e.g., noise) [Rid10]. Typical values of the clock drift are in the order of few parts per million (*ppm*) [Bie99]. For example, a clock drift of 1 *ppm* will respect to a reference clock will cause an asynchrony of over 100 ms after a period shorter than 3 hours. The problem of clock drift can be solved by efficiently using clock synchronization protocols, such as Network Time Protocol (NTP) [Mil10] or Global Positioning System (GPS)[4].

2.6 Impact of Delay Variability and Playout Rate Deviations on IDMS

In this section, the influence of playout rate deviations (mostly driven by clock imperfections) and network delay differences on IDMS is shown.

A receiver with an ideal (or perfect) clock will be able to play out the incoming MUs with exactly the same nominal rate (θ MU/s) as they were generated by the source (see μ_1 in Figure 2.7.a). Nevertheless, due to clock imperfections, the playout rates could present a deviation trend or linear skew, given by $\pm\gamma$ (see μ_2 – slow rate – and μ_3 – fast rate – in Figure 2.7.a), which is typically expressed as a ratio in *ppm.*

[4] The applicability of GPS for time synchronization is explained at the official GPS website: http://www.gps.gov/applications/timing/

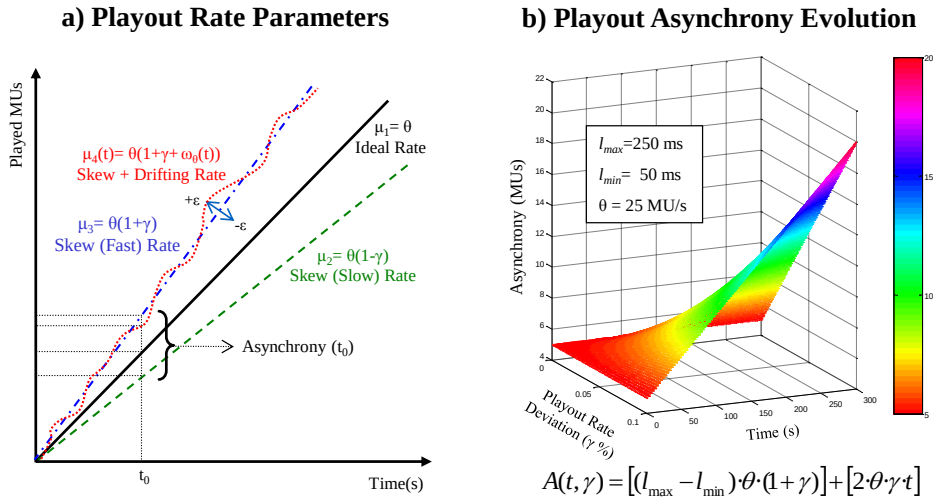


Figure 2.7. Playout Rate Parameters and Asynchrony Evolution.

Moreover, the playout rates could also present a non-linear time-variant drift, given by $\omega(t)$, which is typically modeled as a random fluctuation over the (probably deviated) playout rate, with values bounded by a maximum factor of ± ε *ppm* (see μ_4 in Figure 2.7.a). The playout rate drift also depends on the variable CPU load of the receivers. Reasonable values for playout rate deviations in inexpensive oscillators vary between 10-100 ppm [Fer10]. As a result, the instantaneous playout rate (in MU/s) of the *i-th* receiver can be formulated as:

$$\mu_i(t) = \theta \cdot (1 + \gamma_i + \omega_i(t)) \qquad \textbf{Eq. 2.1}$$

If present, these playout rate imperfections (i.e., skews and drifts) will lead to playout time differences between the different receivers, as their local timing mechanisms will not be in perfect agreement. This will result in different presentation times and durations of MUs in each receiver.

Another factor that can contribute to an initial playout asynchrony between receivers is the network delay differences between the source and each one of them. For instance, if the source begins the transmission rate at t_{ini} instant, and the MUs experience a minimum (maximum) network delay of l_{min} (l_{max}) seconds to reach the nearest (furthest) receiver, and they are buffered the same fixed amount of time (b_{ini}) in all of them, an initial playout time discrepancy or asynchrony will occur, given by $(t_{max}\text{-}t_{min}=(t_{ini}+l_{max}+b_{ini}) - (t_{ini}+l_{min}+b_{ini})=l_{max}\text{-}l_{min})$, being t_{max} and t_{min} the initial playout instants at the furthest and nearest receiver, respectively.

If the playout rate imperfections are also taken into account, the worst case will occur when the nearest receiver plays out the stream at a maximum rate of $\theta \cdot (1+\gamma)$ [5], whereas the furthest one plays out the stream at a minimum rate of $\theta \cdot (1-\gamma)$. This is because once the furthest receiver begins its playout process at t_{max} instant, the nearest one has already played out a certain number of MUs given by $[(t_{max}-t_{min}) \cdot \theta \cdot (1+\gamma)]$ (first term of Equation 2.2). Moreover, the difference between their playout rates ($[\theta \cdot (1+\gamma) - \theta \cdot (1-\gamma)] \cdot t$) will cause an increasing asynchrony, in MUs, between them as the multimedia session advances in time (see Figure 2.7.b). This asynchrony, $A(t,\gamma)$, is given by Equations 2.2 and 2.3 (compacted version) [6], where the first term is the contribution of the delay differences, which can be appreciated in the initial elevation of the graph in Figure 2.7.b (at $t=0$), and the second term represents the effect of the (undesirable) playout rate deviations, γ, and the temporal evolution of the session, t:

$$A(t,\gamma) = \left[(t_{\max} - t_{\min}) \cdot (\theta \cdot (1+\gamma))\right] + \left[(\theta \cdot (1+\gamma)) \cdot t - (\theta \cdot (1-\gamma)) \cdot t\right] \quad \textbf{Eq. 2.2}$$

$$A(t,\gamma) = \left[(l_{\max} - l_{\min}) \cdot \theta \cdot (1+\gamma)\right] + \left[2 \cdot \theta \cdot \gamma \cdot t\right] \quad \textbf{Eq. 2.3}$$

As can be seen from Equation 2.3 and from (the initial elevation of) the graph in Figure 2.7.b, if there is a significant network delay variability between the furthest and the nearest receiver and the same fixed amount of buffering delays are set in both of them, there will be an initial playout asynchrony at the start of the media playout ($t=0$), even without the presence of playout rate deviations (i.e., $\gamma=0$). For instance, if $l_{max}-l_{min}$=200 ms and θ=25 MU/s, we will get an initial asynchrony of:

$$A(t=0,\gamma=0) = \left[(250-50) \cdot 10^{-3} \cdot 25 \cdot (1+0)\right] + \left[2 \cdot 25 \cdot 0 \cdot 0\right] = 5\ MUs$$

Although presentation times are carried in MUs or in the delivery packets, the delay variability between receivers, their heterogeneous computational resources, the different buffer settings, and clock imperfections will lead to playout time discrepancies, which will probably increase as the media session goes on. This behavior is unacceptable in practical shared media experiences. Therefore, proper solutions need to be devised to overcome these challenges and to eliminate the asynchrony between distributed receivers (IDMS). Furthermore, these solutions cannot only be based on buffering techniques, but they need to be adaptive regarding the variability of network conditions, clock' imperfections, as well as the heterogeneity and instability of the processing capabilities of the involved devices.

[5] Here, only the effect of playout rate skews (γ) is considered, as this factor has a bigger impact on the playout asynchrony than playout rate drifts ($\omega(t)$), which oscillate over time, as shown in Figure 2.7.a for μ_4.

[6] $([\theta \cdot (1+\gamma) - \theta \cdot (1-\gamma)] \cdot t) = \theta t + \theta \cdot \gamma \cdot t - \theta t - (-\theta \cdot \gamma \cdot t) = 2 \cdot \theta \cdot \gamma \cdot t$

2.7 Magnitudes of Delays and Delay Differences in Real Scenarios

After presenting the different sources of delay and of delay variability, this section provides insights about the magnitudes of the delay and delay differences in real scenarios, according to numbers found in literature.

The International Telecommunication Union (ITU) G.1050 standard [ITU-T G.1050] reports on typical values of delays and jitter in Internet. It is stated that network delays typically range between 20 and 500 ms, while jitter values are between 0 and 500 ms. Likewise, the ITU-T G.114 standard [ITU-T G.114] indicates that delays lower than 150 ms are required for Internet conferencing, while delays larger than 400 ms are typically unacceptable.

In [Jan11], it is reported that end-to-end delays when using popular videoconferencing systems in a LAN (Local Area Network) scenario range between 99 ms, for Google Talk (Gtalk), and 312 ms (with a standard deviation of 67 ms), for Skype. Likewise, end-to-end delays of approximately 350 ms (with a standard deviation of 67 ms) were measured when using an ad-hoc videoconferencing system in a distributed scenario between Belgium and UK (United Kingdom).

In [Dev08], delay measurements in different content delivery scenarios were performed. A worst-case analysis was made, whose results are in Table 2.1. It can be appreciated that coding, transmission, and buffering are the main sources of video delays. Interestingly, it was pointed out that differences in end-to-end video delays between receivers in an IPTV scenario can be larger than 6 s (with delays ranging between 250 ms and 6500 ms).

Table 2.1. Sources of Delay in Content Delivery Networks (CDNs) [Dev08]

Factor		**Typical Delay Range (ms)**
Source	Video Capture	17 - 40
	Video Encoding	50 - 2000
	Encryption	0 - 50
	Error Protection	0 - 100
	Transmission Buffer	50 - 500
Network	Uplink Transmission	10 - 300
	Transcoding	0 - 2000
	Downlink Transmission	10 - 300
Receiver	Jitter Buffer	50 - 500
	Error Protection	0 - 100
	Decryption	0 - 50
	Video Decoding	50 - 500
	Display Buffer	0 - 50
TOTAL		**250 - 6500**

The research work in [Mek11] provided measurements of playout time differences when receiving the same media content via different TV delivery technologies, such as different DVB variants, analogue cable, IPTV and web-based TV. It was shown that delay differences between broadcast technologies can accumulate up to 5 s, and even up to 8 s when using web-based TV solutions.

Similarly, the study in [Koo14] provided measurements about the magnitudes of delay differences for different TV setups in specific receivers. It was shown that delay differences between different TV broadcasts in a national scenario (in the Netherlands) can accumulate up to almost 5 s, while in an international scenario (between the Netherlands and UK) can accumulate up to 6 s. Likewise, these measurements revealed that analog broadcasts are typically delivered faster than digital broadcasts and that, in general, High Definition (HD) broadcasts are slightly slower than their equivalent Standard Definition (SD) broadcasts. It was also proved that web-based TV solutions can introduce more than 1 minute (up to 70 s) delays compared to "regular" broadcast technologies. In addition, significant delay differences between receivers when using exactly the same TV delivery technology, setup combination (i.e., subscription type/quality) and equipment in all of them were noticed in that study. However, no numbers were provided due to the lack of sufficient measurements from multiple geographically distributed sites and of concluding results.

2.8 Human Perception on Delay Differences

Previous studies have investigated the impact of delay variability thresholds on human perception for different intra-media and inter-media synchronization use cases (e.g., [Ste96], [Hua13]). However, the exact ranges of allowable delay differences (i.e., playout time differences or asynchrony levels that, if exceeded, are noticeable and/or annoying to users) for the different IDMS use cases have not been sufficiently determined yet. Such limits should be obtained through exhaustive and very rigorous objective and subjective assessments (i.e., user perception tests), possibly including longer-term testing in live systems, in contrast to testing in artificial test environments. Likewise, such testing should be performed for each particular IDMS use case under study, as the ranges of tolerable asynchrony levels strongly depend on the usage scenario.

In this context, previous studies have focused on determining the tolerable asynchrony levels in Social TV scenarios. The delay bounds of 150 ms specified in [ITU-T G.114] have been traditionally used as a rule of thumb ([Vai11a], [Gee11]). This rule states that the maximum end-to-end (one-way) delays when remotely communicating should not exceed 150 ms. Below this value users may not perceive the delay in communications. Therefore, it could also be concluded that this delay threshold is also the lower bound for synchronizing shared media content. The study in [Sha08] provided initial evidence that IDMS helps users to feel closer and more

connected when watching video together, while apart. However, no discussion about tolerable asynchrony thresholds was provided. In [Dev08], it was concluded that the bounds on delay differences between receivers to enable interactive video services may range between 15 and 500 ms, depending on the usage scenario. Moreover, controlled experimental setups have analyzed the effect of de-synchronization on the QoE in Social TV scenarios ([Gee11] and [Mek12]). The study in [Gee11] aimed at determining the range of asynchrony levels that are tolerable in Social TV scenarios. In that study, distributed users watched together a quiz show (which is a very sociable genre, according to [Gee08] and [Gee09]), while remotely interacting via voice and text chat. In such testing, various synchronization conditions, with different asynchrony levels, ranging from 0 s to 4 s (in steps of 500 ms), were forced and presented to participants in a randomized order, by enabling one of the two interaction channels (voice and text) in each test. After each test condition, the participants had to fill in questionnaires, asking a series of questions related to togetherness, noticeability and annoyance. It was concluded that playout time differences up to 1 s might not be perceptible by users while communicating using audio conferencing services, but playout time differences above 2 s really became annoying for most of them (i.e., both voice and text chatters). Concretely, when using voice chat, users noticed synchronization differences sooner, were more annoyed and felt more together than when using text chat. However, users with high text chat activity noticed synchronization differences similar to those using voice chat. Similar results were obtained in [Mek12] by recreating a shared football watching experience.

However, the asynchrony thresholds for Social TV are largely dependent on other many factors, such as the genre of the video content ([Gee08], [Gee09]), the number of involved users, their profiles (e.g., age, sex, relationships among them – family, friends, partners –, etc.) and capabilities, their interest in the media content, the (relative) importance of the media content at a specific moment, if the users have watched the video before, the communication channel in use [Hua09], etc. Consequently, no statistically absolute limits may be derived from these preliminary, but relevant, experiments in [Gee11] and [Mek12]. An extension to such studies would be very interesting for determining the impact of such factors and the levels of synchronization that must be provided in real scenarios. Moreover, in [Gee11] a low percentage of participants noticed synchronization differences when setting the lowest delay offset of 500 ms. Accordingly, we believe that more accurate synchronization levels for IDMS in Social TV should be provided, as the worst case is also relevant from the point of view of both customers (to avoid their frustration) and service providers (to avoid complaints of their customers).

Apart from Social TV, approximate asynchrony limits for the different IDMS use cases presented in Section 2.4 are also provided in this Section. Such thresholds have been derived from numbers found in literature, according to physical issues,

related subjective assessments, and opinions of experts from both academy and industry[7].

Table 2.2 gives a preliminary categorization of such IDMS use cases according to the required synchronization levels and to the technical requirements in order of magnitude of the maximum tolerable asynchrony between destinations or consumption devices. The technical requirements are not meant to be exact, but give a qualitative order of magnitude of the maximum tolerable delay differences. These approximations, expressed with intervals and not with exact values, are derived from the functional reason for synchronization:

- *Very high synchronization* (asynchronies lower than 10 ms) is necessary for different audio outputs in a single physical location. For example, this is necessary for proper sound localization, as explained in [Pit10]. That work explains about audio localization and the granularity of the human ear, which can recognize differences of 10 μs, or even lower, between the arrival times of sound at each ear.
- *High synchronization* (asynchronies between 10 and 100 ms) is required for any use case in which fairness between the involved users is important. Typical response times of users should not be influenced too much by delay differences of media playout to which users respond. As explained in [Nie93], 100 ms is a well-known upper limit for users to feel that a system is reacting instantaneously. Likewise, as reported in [Mau04] and in [Roc08], delay thresholds around 150-200 ms are typically desired to keep an enjoyable shared experience in MOGs. Likewise, synchronization mechanisms are needed to ensure a consistent global view of the state of the game. Therefore, the synchronization mechanisms to be implemented need to guarantee that both the degree of interactivity and delay differences between distributed players are below these thresholds.
- *Medium synchronization* (asynchronies between 100 and 500 ms) is required in cases in which various related media items are displayed somewhat simultaneous, but in which no tight real-time requirements are posed. Typical use cases at this level are about semi real-time additional content, or about users who are consuming content at different physical locations and do have active interaction, but not so strict as in the high-accuracy scenario. For such interactive sessions, the delay should be kept in limits where it does not impact (conversational) dynamics too much, typically within the order of several hundred milliseconds, as explained in [ITU-T G.1010]. Likewise, the study in [Dev08] categorizes the ranges of

[7] The opinions or guidelines from experts regarding the required synchronization levels were obtained through discussions during the standardization process of our IDMS solution (presented in Chapter 8) within the IETF and through individual interviews.

asynchrony levels for different interactive TV services between 15 and 500 ms. In that study, it is also stated that asynchrony levels around 100 ms may already be noticeable and annoying in various IDMS use cases.

- *Low synchronization* (asynchronies between 500 and 2000 ms) is required in cases where media is consumed by different users at different physical locations, but the interaction level between them is not of a very competitive nature. The users' perception studies in [Gee11] revealed that 500 ms is the lower threshold at which (a low percentage of) participants start to actually notice synchronization differences in Social TV. Furthermore, it was shown that asynchrony levels above 2 s really become annoying for most users, independently on the interaction channel in use. Accordingly, an upper bound of 2 s delay differences has been chosen in the low synchronization range.

Table 2.2. Classification of IDMS Use Cases

Synchronization Level	Technical Requirement	Relevant use cases
Very high	10 µs – 10 ms	- Networked stereo loudspeakers - Phased array transducers - Video wall
High	10 – 100 ms	- Multi-screen settings - Distributed tele-orchestra - Networked quiz shows - Networked real-time multi-player games (MOGs) - Multi-party conferencing - Conferencing sound reinforcement system - Game-show participation
Medium	100 – 500 ms	- Synchronous e-learning - Synchronous Groupware - Presence based games - Consumer-originated content - On-line election events
Low	500 – 2000 ms	- Seamless switching among media devices - Shared service control - Social TV

This analysis and categorization reveals that the tolerable asynchrony levels for IDMS are much lower than the magnitudes of delay differences in current-day delivery scenarios (discussed in Section 2.6). This is true even when considering the "soft" synchronization levels required in Social TV, which is a very relevant use

case. Consequently, it can be concluded that current delivery platforms do not handle the IDMS problem in an optimal way, and that actual delay differences may prevent the inclusion of advanced forms of interactivity in group shared media experiences, leading a severe QoE degradation, as empirically proved in [Gee11] and [Mek12]. This motivates the design of adaptive and accurate IDMS solutions (as well as of other types of multimedia synchronization) to compensate for such end-to-end delay variability, which is the goal of this PhD thesis.

2.9 Summary

This Chapter has provided an in-depth introduction to the research area covered in this PhD thesis. The existing types of temporal multimedia synchronization have been classified and their relevance has been discussed. One of the goals of this classification was to provide a general view of how IDMS, which is the particular synchronization type this PhD thesis is focused on, fits in the overall multimedia synchronization research space. The wide applicability and relevance of IDMS has been discussed, by compiling up to 20 use cases in which this type of synchronization is necessary or useful.

This Chapter has also provided an overview of the different sources of delay and of delay variability when delivering media over distributed scenarios. A proper understanding of the origin of these sources of delay and of delay variability, as well their integrated treatment, is essential to devise proper multimedia synchronization solutions. In relation to that, it has been shown that the magnitudes of the delay differences when delivering media over distributed scenarios significantly exceed the allowed thresholds in the compiled IDMS use cases. Therefore, this reveals the need for designing the appropriate technology to efficiently provide IDMS (thus compensating for such end-to-end delay variability), which is the main goal of this PhD thesis.

Chapter 3

IDMS COMPONENTS AND CLASSIFICATION

3.1 Introduction

In order to design and deploy a complete IDMS solution, an integration and/or interaction between several components (e.g., entities, protocols, architectural schemes, algorithms, techniques…) is necessary. In this Chapter, the various components of an IDMS solution are identified, and different options for several of such components are presented. Likewise, key concepts about IDMS are introduced, and synonyms for each one of them are also identified (as different terms are commonly used in literature to refer to the same or a similar concept). This will enable the use of a consistent terminology all over the memory. Finally, an overview of quality metrics to evaluate the IDMS performance is provided.

3.2 Components of an IDMS Solution

An IDMS solution is typically comprised of the following main components:

- Involved entities in the media delivery and IDMS processes (presented in Section 3.3).
- Protocols for media delivery and for exchanging useful information to achieve IDMS.
- Clock synchronization mechanisms (if any).
- Protocols for session management and negotiation of key features for IDMS (e.g., encoding settings, clock references, groups' establishment…).

- Architectural schemes between the involved entities to exchange information about IDMS (different possibilities in this aspect are presented in Section 3.4 and in Section 3.5).
- Control algorithms (e.g., monitoring and calculation of playout asynchronies, selection of a reference playout timing to synchronize with, fault tolerance…).
- Control or adjustment techniques to maintain and/or restore synchronization (different possibilities are presented in Section 3.6).

The selection of a particular architectural scheme will determine the involved entities and their communication processes to achieve IDMS. However, this decision can be independent of the control algorithms and adjustment techniques to be employed. Figure 3.1 aims to clarify these relationships. Note that this figure is not meant to encompass all possible cases, but to provide a general idea about such IDMS components and their interactions. For instance, different entities can be involved in a specific IDMS solution (explained in Section 3.3), which can be placed at different locations and can follow different architectural schemes (explained in Section 3.4 and Section 3.5). Furthermore, different control algorithms and adjustment techniques can also be implemented in different entities (explained in Section 3.6).

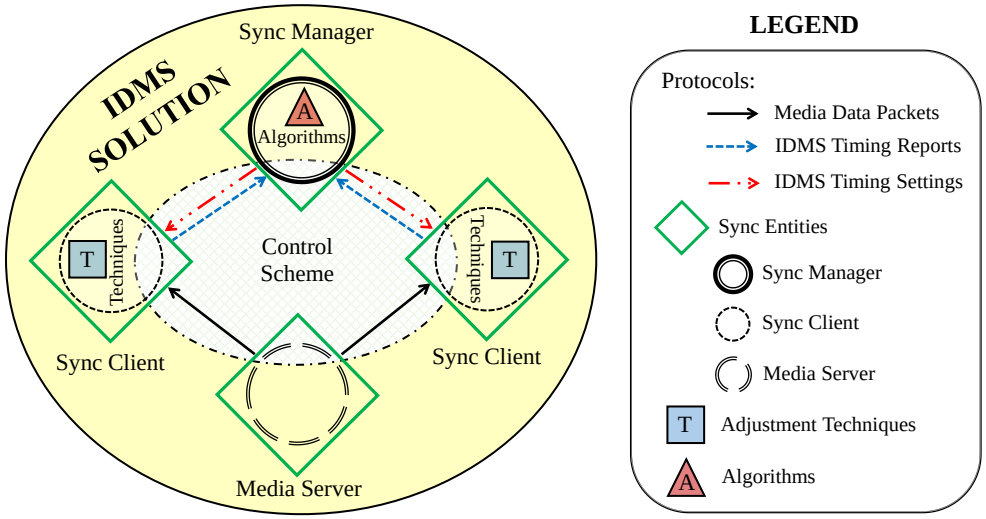


Figure 3.1. IDMS Components and their Relationship.

3.3 Involved Entities in the IDMS Process

Different (sync) entities can be involved in the IDMS control process (see Figure 3.1), each with a particular role:

- *Media Server*: It is the sender of the media stream. There can be a single or multiple Media Servers transmitting one or several media types and/or streams, as discussed in Section 2.2. Other terms, such as server, sender, transmitter and source are also commonly used to refer to this entity.
- *Sync Clients*: They are the sync entities that need to render the media content in a synchronized manner. The Sync Clients typically have to send informative feedback reports, including their current (reception and/or playout) timing information, to allow for an overall IDMS control. Other terms, such as client, receiver, participant and destination are also commonly used to refer to this entity.
- *Master Sync Client*: It is a specific Sync Client whose (playout) timing information is selected as the IDMS reference for adjusting the playout timing of the other (slave) Sync Clients.
- *Sync Manager*: It is the sync entity responsible for collecting the IDMS reports from the Sync Clients. Then, it must calculate the timing discrepancies between the Sync Clients and, if needed, send new control messages back to them including setting instructions to achieve IDMS. Such control messages typically include a target playout point for all the Sync Clients, which could be based on the reported playout timing by a selected Master Sync Client. Other terms, such as MSAS (Media Synchronization Application Server), maestro, and synchronizer are also commonly used to refer to this entity.

3.4 Architectural Approaches for IDMS

When implementing IDMS, the first architectural decision is to determine the most appropriate location for the sync entities. Two main approaches can be followed: *network-based* and *terminal-based*.

In *network-based approaches* (Figure 3.2.a), the sync entities (i.e., the Sync Manager and Sync Clients) are deployed at the network side, and the IDMS control processes are managed by those network entities, under control of the service provider or the operator. This way, the end users' terminals do not have to implement any IDMS functionality. In *terminal-based approaches* (Figure 3.2.b), the Sync Clients are located at the end-users' terminals, while the Sync Manager can be deployed as a separate independent entity, or either as part of a Sync Client or of a Media Server (as shown in Figure 3.3.b). In both approaches, the Sync

Manager and the Sync Clients exchange IDMS control messages to guarantee an overall synchronization status in the group shared media experience.

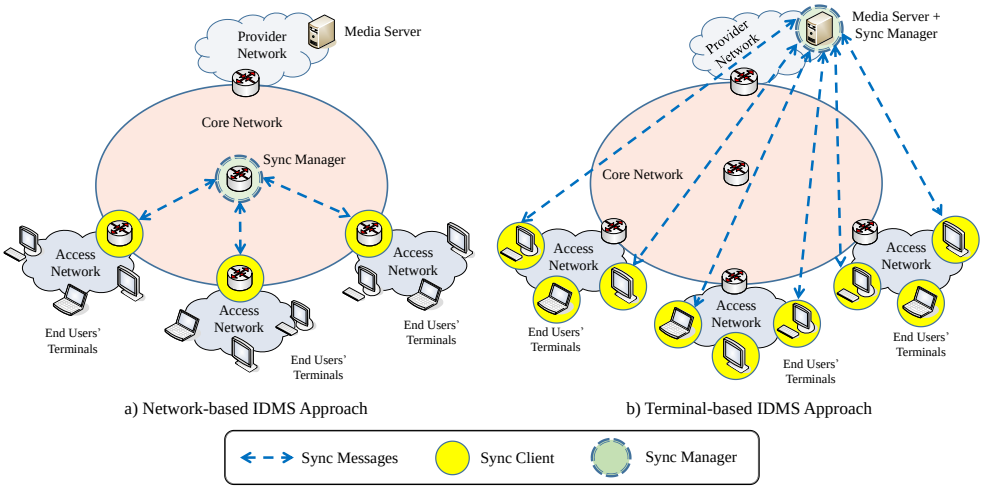


Figure 3.2. Classification of IDMS based on the Functionality Location.

Network-based IDMS solutions have been neither deployed nor tested in any real scenarios yet. Only a design approach was proposed in [Sto10] to meet the need of IDMS in advanced and large-scale IPTV services. Accordingly, the taxonomy of IDMS solutions provided in this thesis (in Chapter 4), as well as the designed IDMS solution (in Chapter 8), are mainly focused on terminal-based approaches.

3.5 Control Schemes for IDMS

Independently of the architectural approach, three structural or control schemes to perform IDMS have been identified in literature, depending on the distribution of the sync entities and on their communication processes for exchanging information about IDMS (see Figure 3.3): two centralized schemes, Master/Slave (M/S) Scheme and Synchronization Maestro Scheme (SMS); and one distributed scheme, Distributed Control Scheme (DCS).

In M/S Scheme (Figure 3.3.a), the Sync Clients are classified into master (one of them) and slave (the rest). Only the master Sync Client is responsible for sending (typically in a multicast way) IDMS reports (including useful timing information to achieve IDMS) to the rest of (slave) Sync Clients. Accordingly, each slave Sync Client will adjust its own playout process to synchronize with the reference IDMS timing reported by the master Sync Client. The master Sync Client can be assigned in an arbitrary manner or according to specific priority roles (e.g., the teacher's client

application in e-learning applications, or the doctor's client application in e-health applications).

SMS (Figure 3.3.b) is based on the existence of a centralized Sync Manager, which could be a completely separate sync entity (Figure 3.3.b.1), the Media Server (Figure 3.3.b.2), or one of the Sync Clients (Figure 3.3.b.3). In this centralized scheme, the distributed Sync Clients send (typically in a unicast way) IDMS reports to the Sync Manager. Once the Sync Manager has collected the IDMS reports from all the Sync Clients, if it detects an asynchrony situation between their playout processes exceeding an allowed (pre-specified) threshold, it will send (typically in a multicast way) a new control message to the distributed Sync Clients including IDMS setting instructions. Accordingly, the Sync Clients will have to enforce the required playout adjustments to achieve IDMS.

Unlike M/S Scheme, which only requires a unidirectional communication process (between the master and slave Sync Clients), SMS requires a bidirectional communication process (between the Sync Manager and all the Sync Clients).

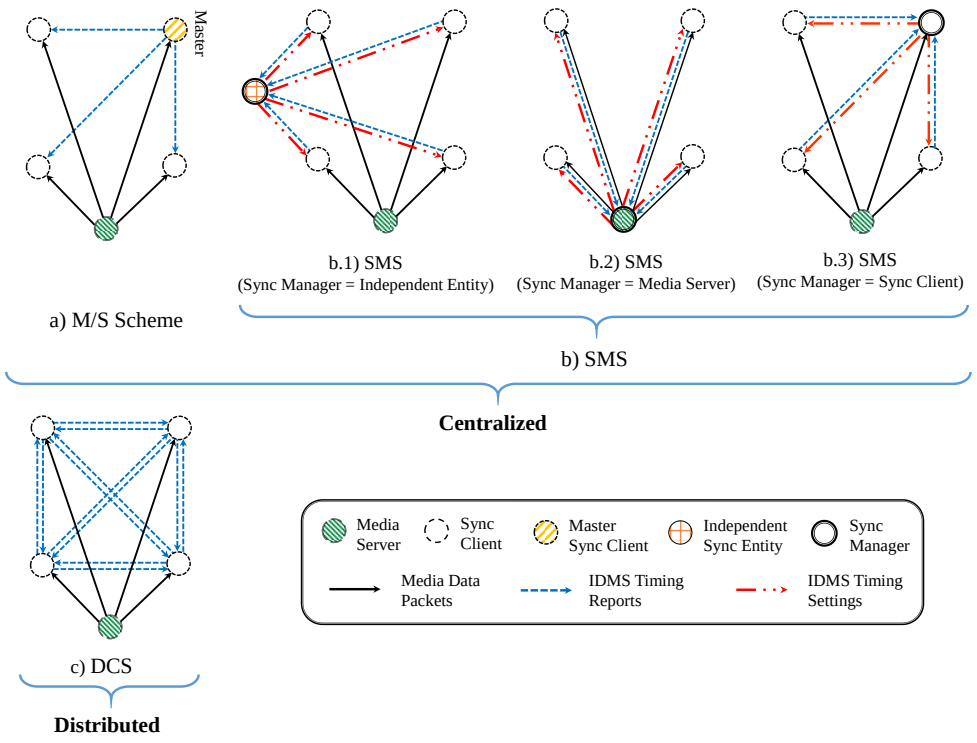


Figure 3.3. Control Schemes for IDMS.

Moreover, in SMS, the Sync Clients can also be classified into an M/S Scheme, in which the IDMS timing reported by a master Sync Client will be chosen as the reference for adjusting those of all the other (slave) Sync Clients. Likewise, the master Sync Client role can also be dynamically switched between all the Sync Clients according to the variable network and end-systems conditions [Bor09a].

Finally, in DCS (Figure 3.3.c), all the distributed Sync Clients exchange (typically in a multicast way) IDMS reports. Therefore, each Sync Client will have a global knowledge of the synchronization status in the shared session and can locally decide the IDMS reference to synchronize with from among its own IDMS timing and the ones received from the other Sync Clients.

A hybrid architecture, based on a combination of both SMS and DCS can also be used. Herein, the Sync Clients can be divided in sub-level domains, each one with a local Sync Manager for controlling the IDMS timing of all the Sync Clients in that domain. On a multi-domain level perspective, the different local Sync Managers can further communicate with a higher hierarchic Sync Manager that is responsible of controlling the IDMS processes of the associated sub-level domains. A similar approach is utilized for managing the consistency of the session in MOGs ([Roc08], [Fle10]). However, even though acknowledging its relevancy for enabling large-scale scenarios (e.g., IDMS in IPTV environments), the design and evaluation of this hybrid architecture is not covered in this PhD thesis.

As a proof of relevance, Table 3.1 includes a classification of existing IDMS solutions based on both the architectural approach and control scheme in use. In Chapter 4, such IDMS solutions will be described and classified more exhaustively.

Table 3.1. Classification of IDMS Solutions based on the Adopted Control Scheme and Architectural Approach

		Control Scheme		
		Centralized		**Distributed**
		M/S Scheme	**SMS**	**DCS**
Architectural Approach	**Terminal-Based**	[Esc94], [Vai11a], [Ish97], [Gee11], [Sha12b], [Hua12]	[Ish97b], [Tas02], [Ish03], [Ish04], [Has06], [Kur07], [Bor08], [Bor09c], [Sha12b]	[Aky96], [Ish99], [Dio99], [Ish02], [Mau04], [Cro04], [Pal04], [Hos09], [Vai11a], [Rai14]
	Network-Based	-	[Sto10]	-

3.6 Overview of Control or Adjustment Techniques for IDMS

A huge variety of potential control and/or adjustment techniques to achieve IDMS (as well as other types of multimedia synchronization) have been devised. These adjustment techniques are considered as unique and indivisible or atomic (with no

different functions from the multimedia synchronization point of view) since any one of them, either alone or in combination with others, can be employed to achieve the synchronization goal.

In line with the studies in [Ish00] and in [Bor09a], the synchronization adjustment techniques can be classified into four main categories according to their purpose (basic, preventive, reactive, and common adjustment techniques), which, in turn, can be divided into two groups, depending on the location at which they are executed (server or client side). If the adjustments are performed at the server side, feedback information about IDMS will be typically required from the clients. Likewise, the use of server-based adjustment techniques requires a coordination with client-based techniques, as each individual Sync Client has to perform the required adjustments to achieve IDMS.

This section provides a classification and brief description of the most relevant adjustment techniques in each category and location, whose summary is presented in Table 3.2. The goal is not to compare such techniques, but to identify the possibilities to achieve IDMS and their applicability for specific use cases. The implementation of such techniques in existing IDMS solutions will be explored in Chapter 4.

3.6.1 Basic adjustment techniques

Basic adjustment techniques are needed in most of the multimedia synchronization solutions, and are essential to preserve the temporal relationships within or among streams in shared media experiences.

- **Server-based techniques**. Basic adjustment techniques executed at the server side typically consist of *adding some information useful for synchronization* in (the headers of) the data packets (encapsulating MUs), such as payload type identification, timestamps, sequence numbers, markers, event information, source identifiers and group identifiers. The *payload type* information allows identifying *on-the-fly* the type of media and encoding settings of the incoming MUs. This information is important, since it can determine the targeted playout rate and the required settings of the decoding processes. *Timestamps* can contain the generation instants, as well as the targeted decoding and presentation instants, of each MU. Hence, they are very useful to schedule media playout. *Sequence numbers* are useful to reconstruct the original order of incoming MUs and detect losses. It is important to note than when the generation of MUs is periodic, the use of sequence numbers and payload type information can suffice for reconstructing the timing of the incoming media streams, without the need of timestamps. *Markers* can be used to time align the playout processes of different media streams, while *event information* can be used to trigger synchronization adjustments at specific instants. *Source identifiers* are

useful to associate incoming streams from the same or related Media Server(s), and *group identifiers* are needed to allow an independent IDMS control for different logical groups of Sync Clients.

- **Client-based techniques**. Nearly all the existing synchronization solutions use *playout buffering techniques* at the client side. The playout buffers store the incoming MUs a sufficient period of time to compensate for the effect of jitter. The presentation times of MUs will be given by a trade-off between the magnitude of jitter and the synchronization information of each incoming MU, with the final goal of reconstructing the original timing of the incoming media streams, while minimizing the latency of the multimedia service.

Table 3.2. Control or Adjustment Techniques for IDMS (adapted from [Bor09a])

Technique's purpose	Location	Technique
Basic (B)	Server (S)	Addition of useful metadata for synchronization (timestamps, sequence numbers, source and group identifiers, markers, event information, etc.).
	Client (C)	Buffering techniques.
Preventive (P)	Server (S)	Initial playout instant calculation.
		Deadline-based transmission scheduling.
		Interleave MUs of different media streams in a single transport stream.
	Client (C)	Preventive skips of MUs (eliminations or discardings) and/or preventive pauses of MUs (repetitions, insertions or stops).
		Adjustment of the buffering time of MUs.
Reactive (R)	Server (S)	Adjustment of the transmission timing.
		Decrease the number of transmitted media streams.
		Drop low-priority MUs.
	Client (C)	Reactive skips (eliminations or discardings) and/or reactive pauses (repetitions, insertions or stops).
		Playout duration extensions or reductions.
		Use of a virtual time with contractions or expansions.
		Master/Slave switching.
		Late events discarding (Event-based).
		Rollback techniques (Event-based).
Common (C)	Server (S)	Skip or pause MUs in the transmission process.
		Advance the transmission timing.
		Adjustment of the input rate.
		Media Scaling
	Client (C)	Adjustment of the playout rate.
		Data interpolation.

3.6.2 Preventive adjustment techniques

Preventive adjustment techniques attempt to avoid situations of asynchrony, before they occur.

- **Server-based techniques**. For stored media content, the server can schedule the transmission of MUs according to some *synchronization information* (for example, timestamps), in order to meet the targeted deadline requirements. For that purpose, the size of each MU, its playout deadline and the network delay bounds (or, at least, the Probability Distribution Function or PDF of the delay) should be known. This technique is commonly known as *deadline-based transmission scheduling.*

  The Media Server can also use a technique based on *interleaving* MUs of different media streams in a single transport stream. This technique can improve the inter-media synchronization quality in shared media experiences (as the delay differences between streams are eliminated), but may degrade the intra-media synchronization quality in those streams extremely sensitive to network jitter.

  Likewise, the Media Server can prevent from an initial asynchrony situation by scheduling a *global initial transmission and/or playout instant* for all media streams and Sync Clients, respectively, at the beginning of the session. The global initial playout instant will need to be communicated to all the Sync Clients before, or in parallel with, the transmission of the first MUs.

- **Client-based techniques**. In some cases, the Sync Clients could perform *preventive skips* of MUs (eliminations or discardings) and/or *preventive pauses* of MUs (repetitions or insertions) depending on the fullness level of their playout buffers. In some cases, it can also be possible to *insert dummy* (noise) *data*, instead of pausing (or stopping) the playout processes. When using multi-level coding systems, Sync Clients can also *discard MUs with lower priority* (e.g., B frames in MPEG), according to their playout buffers occupancy or playout deadline requirements

  If the Sync Clients are able to somehow estimating the network delay experienced by MUs, they can also change their waiting time at the playout buffer. It is also possible to *enlarge or shorten the silence periods in audio streaming* on a talkspurt-by-talkspurt basis (e.g., by employing a speech activity detector that classifies a voice signal as either talking or silent). However, this technique is not suitable when streaming music, as the periods of silences are almost as important as the periods of sound in such a case.

3.6.3 Reactive adjustment techniques

Reactive adjustment techniques are used to recover synchronization after the detection of an asynchrony situation.

- **Server-based techniques**. On the one hand, if the Media Server is able to know the existing playout asynchrony between the Sync Clients, it can *adjust the transmission rate* to allow deviated (especially lagged) Sync Clients achieve IDMS. On the other hand, if the Media Server detects that synchronization is difficult to achieve by specific Sync Clients, it can *decrease the number of transmitted media streams*. For example, in multi-party conferencing, if an asynchrony is detected and this situation persists, the Media Server could stop the transmission of the video stream temporarily, only maintaining the audio transmission. Accordingly, when the Media Server detects that synchronization has been re-established, it will restart the transmission of the video stream. Besides, when using multi-level coding systems, the Media Server can *drop lower priority MUs* (for instance, B frames in MPEG) according to some QoS parameters (such as the network congestion or loss rate). This may also help to improve the synchronization performance.

- **Client-based techniques**. Sync Clients can perform several reactive techniques to recover from asynchrony situations. The most popular technique, due to its easy implementation, consists of *reactive skips* (eliminations or discardings) and/or *reactive pauses* (repetitions or insertions). For example, if a specific Sync Client detects that the scheduled presentation time of the MU it is processing has already expired, it can choose between to play out that MU and discard the consecutive one/s (that were received before), as in [Ish97a], or to discard it directly, as in [Man06] and [Bor08]. In audio streams, a solution to deal with loses and delayed MUs can be to simply stop the playout process [Man06]. Likewise, when MUs of different streams are sent at the same instant (and have the same timestamp), a common technique consists of pausing the playout of the MUs of a specific media stream until the associated MUs of the other streams are received.

  In lagged streams, the playout process can be blocked or suspended (*blocking policy*) until late MUs (i.e., MUs that do not arrive at the Sync Client in time to meet their respective playout deadlines) arrive. In specific cases, the playout process is only blocked for a pre-specified period of time (*restricted blocking policy*). In other cases, when buffer underflow situations occur, the Sync Client can opt to *repeatedly play out the last MU* until the next one is available. It would be also possible to play out other data (e.g., scene information) at that moment.

However, experience has demonstrated that abrupt (or aggressive) playout adjustments, such as skips and pauses, can result in long-term playout discontinuities or disruptions, which are annoying to users (e.g., [Ish03b, [Su09], [Hss19]). To overcome this issue, *Adaptive Media Playout (AMP)* techniques can be used. Such techniques consist of *adjusting the media playout rate* (i.e., playing the media faster/slower than normal), within perceptually tolerable ranges, to recover from undesired situations (e.g., buffer underflow/overflow or asynchrony situations), while providing glitch-free audio-visual quality. For video, the Sync Client has to simply adjust the display duration for each video frame. Nevertheless, AMP for audio is less straightforward than for video. In this case, the Sync Client has to perform signal processing in conjunction with time scaling techniques to stretch or widen an audio sequence, while preserving the pitch of the signal. For voice streams, a speech activity detector can be used to shorten or extend the silence gaps.

AMP techniques were originally targeted for improving the intra-media synchronization performance, but can also be used to adjust the playout timing of a specific media stream to that of another stream (i.e., to achieve both inter-media synchronization and IDMS).

The extension of the playout duration is a similar technique to the use of preventive pauses, because the latter can be performed by enlarging the playout duration of specific MUs. Nevertheless, the former is a reactive technique while the latter is preventive.

Another reactive client-based technique is the *use of a virtual time with contractions or expansions* to achieve the synchronization goal, as in [Ish97a]. In this technique, the MUs are played out using a virtual time axis different from the real time axis. Virtual time expands or contracts the service time of incoming MUs according to the amount of jitter experienced by them. This technique differs from AMP (i.e., shortening and extension of playout duration) in that the former indirectly changes the playout timing by modifying the virtual-time (i.e., re-setting the origin of the time axis), while the latter directly changes it (i.e., the origin of the time axis is kept the same).

When using M/S Scheme for IDMS, if the playout timing of the master Sync Client is extremely deviated (lagged or advanced) due to some trouble, the *master/slave roles could be dynamically switched* to improve the IDMS efficiency and/or the fairness among the participants. Similarly, a specific Sync Client can *switch the roles of the master and slave streams (inter-media synchronization)*, according either to their global importance in a group shared media experience or to their playout time discrepancy.

Two other reactive techniques are commonly employed in networked games: *discarding of late events*, as in [Dio99]) and *rollback techniques*, as in [Mau04]. The former aims to prevent that late events have an impact on the interactivity of the multimedia service, while the latter is used to re-establish the current state of the session to an older one if an inconsistency is detected.

3.6.4 Common adjustment techniques

Finally, other *common* adjustment techniques can be used as a means to prevent (preventive) or correct (reactive) situations of asynchrony.

- **Server-based techniques**. According to feedback information from the Sync Clients, the Media Server can *skip or pause specific MUs* in the transmission process in order to prevent from asynchrony situations and/or facilitate their correction. Moreover, the Media Server could send dummy/empty MUs, instead of skipping, when the generation rate is lower than the transmission rate. The insertion of dummy data, however, may increase the network load and the end-to-end delay.

  In CoD services, the Media Server can also dynamically *advance the transmission timing*, depending on the network delay estimations. This technique differs from the deadline-based transmission scheduling technique in the following point: the former dynamically schedules the transmission of MUs, while the latter statically schedules it. This technique is also different from the adjustment of the transmission timing because the schedule of the transmission of MUs is performed according to the network delay estimation in the first one, while that is done according to the asynchrony between the Sync Clients in the second one.

  Another technique consists of the *adjustment of the input rate*, by means of varying the clock frequency of the input device, and of the *sampling rate*, according to the synchronization status. Also, *data interpolation* at the Media Server side can be used to adjust the effective input or transmission rate.

  Media Scaling is another useful procedure. The Media Server can dynamically adjust the video (temporal or spatial) resolution according to the overall synchronization status. Also, in layered media encoding, more or less media streams can be transmitted depending on both the synchronization status and the network conditions.

- **Client-based techniques**. One of the techniques included in this group is the *adjustment of the playout rate* by modifying the clock frequency of the playout devices (i.e., hardware clock rate), according to the synchronization

status. *Data interpolation* techniques at the Sync Client side to adjust the effective playout rate can also be useful to achieve synchronization.

3.6.5 Discussion

In general, several adjustment techniques, of different categories, can be employed in IDMS solutions. For example, since preventive adjustment techniques cannot completely avoid asynchrony situations, the combination with reactive adjustment techniques is needed. Similarly, server-based and client-based techniques are also typically used together, as Media Servers and Sync Clients need to cooperate to achieve the synchronization goal. On the one hand, server-side techniques need feedback information from the Sync Clients, or from the network, to let the Media Server to calculate the synchronization status, and to proceed consequently. On the other hand, client-based techniques are necessary due to network dynamics (e.g., jitter) and to perform the required adjustments. A typical example is the combination of the addition of useful information for synchronization in the headers of the data packets (e.g., sequence numbers, timestamps, identifiers…) at the server side, with buffering techniques, and either reactive skipping and pausing or AMP techniques at the client side. Furthermore, several techniques for the same purpose can be simultaneously used at the same location (e.g., several reactive control techniques at the client side). Some other techniques cannot, however, be used cooperatively, such as adjustment of the playout rate and interpolation of media data.

3.7 Quality Metrics to Assess the IDMS Performance

Numerous objective and subjective quality metrics can be used to evaluate the performance of IDMS solutions (being most of them also applicable to other multimedia synchronization types). Table 3.3 includes a list of the most relevant metrics, together with their formulation and a brief description.

This overview of assessment metrics, in conjunction with the alternatives of the components previously presented in this chapter, will be used to classify the IDMS solutions (in Chapter 4).

Table 3.3. Metrics to Assess the IDMS Performance

Metric	Formula	Description
Average Playout Rate (AvgP)	$AvgP = \frac{1}{N}\sum_{n=0}^{N-1} \mid \mu_n \mid$	Average number of MUs played out per time interval (e.g., per second) by a specific Sync Client.
Mean Discrepancy From the Original Playout Rate (MdP)	$MdP = \frac{1}{N}\sum_{n=0}^{N-1} \mid \theta - \mu_n \mid$	Average deviation between the original Media Server rate (θ) and the local playout rate (μ) of each Sync Client.
Standard Deviation of the Playout Rate (StdP)	$StdP = \sqrt{\frac{N\sum_{n=0}^{N-1}[\mu_n]^2 - \sum_{n=0}^{N-1}[\mu_n]^2}{N\cdot(N-1)}}$	Similarly to *AvgP* and *MdP*, *StdD* metric is commonly used to evaluate the smoothness of the playout process ([Su09]) in each Sync Client.
Coefficient of Variation of Playout Interval	$\frac{StdP}{AvgP}$	*StdD* divided by *AvgP*. It is used to evaluate the smoothness of the playout process.
MU discard/loss rate	$\frac{MU_{discarded\,/\,loss}}{MU_{generated}}$	Ratio of the number of MUs or synchronization events that have been either discarded (because of late arrival) or lost, to the total number of MUs generated by the Media Server.
Number and Magnitude of Rollbacks	-	Such metrics are usually employed in MOGs. They indicate the total number of rollbacks (i.e., re-establishments of the current state of the session to an earlier one if an older event than the last one being executed is received) and their magnitude (i.e., how long the session state has been incorrect). Likewise, the total number of re-executed commands due to rollbacks is also typically assessed.
Inconsistency rate	$\frac{MU_{inconsistency}}{MU_{generated}}$	Ratio of the number of MUs that have not been received by all the Sync Clients to the total number of sent MUs by the Media Server.
Reversal rate	$\frac{MU_{out\_of\_order}}{MU_{generated}}$	Ratio of the number of MUs which are played out in different order from the generation order (i.e., out-of-order MUs) to the total number of MUs played out by each Sync Client. For example, when a Sync Client plays out the *n-th* MU immediately after the *(n+1)-th* MU, the number of MUs which are played out out-of-order is increased by two.

Average MU delay (AvgD)	$AvgD = \frac{1}{N}\sum_{n=0}^{N-1} \mid p_n - t_n \mid$	Average time difference between the transmission instant, t_n, of each MU and their playout instant, p_n, at each Sync Client. In some works (e.g., [Dio99]), the distribution of the delay experienced by MUs, the standard deviation, as well as the percentage of late and loss MUs, are also measured.
Synchronization Delay * * In some works this metric is also known as: Response Time [Ish02], Total Execution Delay [Cro04], Game Time Difference (GTD) [Pal04], and Response Delay [Hua12].	-	Time interval between the instant at which an event is triggered or a control packet is sent, and the instant at which the corresponding action is executed at the target side. This metric is often used in MOGs, in which consistency and interactivity are crucial aspects. In interactive scenarios, the round-trip delay is typically also considered. The maximum, minimum, average and standard deviation of this metric, as well as its Cumulative Distribution Function (CDF), are also commonly assessed. Likewise, this metric can be compared with a specific threshold to try to prevent the loss of interactivity (e.g., by discarding obsolete events).
Evolution of the End-to-End or Playout Delay	-	Evolution of the end-to-end or playout delay during the media session.
Total Paused MUs and Pause Time	-	Such metrics denote the total number of paused MUs and the accumulated pause time, respectively, during the media session due to the required synchronization adjustments.
Total Skipped MUs	-	It denotes the total number of skipped MUs during the media session due to the required synchronization adjustments.
Number of Discontinuities and Total Discontinuity Time	-	The number of discontinuities counts the times that a specific *i-th* Sync Client has no MUs to play out (i.e., the number of buffer underflow occurrences), while the total discontinuity time represents the accumulative time of all discontinuities the *i-th* Sync Client suffers during the media session.

Buffer Fullness Variation	-	It denotes the increase/decrease of the playout buffer occupancy (measured in MUs, seconds, or even in bytes) at each Sync Client during the media session due to the required synchronization adjustments.
Thresholds of the Playout Rate Variation	-	When using smooth playout adjustments to achieve synchronization, the percentage, frequency, as well as the upper bounds, of the playout rate variation (i.e., fast up or slow down) are commonly assessed.
Mean Square Error (MSE) and Root MSE (RMSE) of the Playout Times	$MSE_{IDMS} = \frac{\sum_{n=0}^{N-1}(p_n^i - p_n^j)^2}{N}$ $RMSE_{IDMS} = \sqrt{\frac{\sum_{n=0}^{N-1}(p_n^i - p_n^j)^2}{N}}$	The MSE_{IDMS} refers to the average square of the difference between the playout times of each MU, excluding skipped MUs, at two given (*i-th* and *k-th)* Sync Clients. This metric reflects the fairness between the Sync Clients. Lower values of MSE_{IDMS} mean better IDMS performance. The Root MSE of IDMS ($RMSE_{IDMS}$) can also be assessed. In a shared media experience, several Sync Clients are typically involved. Therefore, many combinations between each pair of Sync Clients should be considered when assessing the performance in terms of this metric. However, some combinations can have the same tendency as other combinations or have very small differences. The common approach is to select a reference Sync Client (e.g., the most lagged or advanced one) and then evaluate the MSE_{IDMS} between its playout process and that of the other Sync Clients.
Asynchrony Evolution	-	Asynchrony (i.e., playout time differences) between specific Sync Clients (e.g., between the most lagged and most advanced ones).
Maximum, minimum and mean value of the Asynchrony	-	
CDF of the Detected Asynchrony	-	

Traffic Overhead	-	It refers to the extra traffic added to the media session due to the required exchange of information about IDMS.
Computational Load	-	It is often measured as the increase of CPU load due to the IDMS control. It is affected by the number of messages sent or received, and the processing load due to the executed IDMS algorithms and adjustment techniques.
Perceptual Evaluation of Speech Quality (PESQ)	Score ranging from 1 to 4.5, where larger values mean that the (degraded) audio signal is more approximate to the reference, and hence, a better audio intelligibility.	PESQ is a metric for an objective assessment of the quality of audio signals. It is defined in ITU-T P.862 [ITU-T P.862] and provides an automatic computation of the quality of a (degraded) audio signal during the presence of the original reference signal.
Mean Opinion Score (MOS)	Five Scale Metric for the Asynchrony Perception Ranges: 5) Imperceptible; 4) Perceptible, but not annoying; 3) Slightly Annoying; 2) Annoying; and 1) Very Annoying.	Unlike the above metrics which are objective metrics, MOS is a commonly used metric for subjective assessments of the IDMS performance through user perception tests.

3.8 Summary

This Chapter has introduced the necessary components to constitute a complete IDMS solution, as well as the interaction between them. First, the main entities involved in the IDMS control process have been presented. Next, the architectural approaches and control schemes that have been adopted up to date to perform IDMS have been identified, by explaining the involved entities in each one of them, and the required communication process between them to achieve IDMS. After that, the most common control and/or adjustment techniques to achieve IDMS have been compiled. Moreover, such techniques have been classified according to their purpose (basic, preventive, reactive, and common adjustment techniques) and to the location at which they are performed (at the server and/or client side). Finally, an overview of quality assessment metrics for IDMS has been provided.

The content of this Chapter is useful to: i) better understand the research topic and contributions of this PhD thesis; ii) identify the necessary components and options to deploy an IDMS solution; and iii) perform a classification of the existing IDMS solutions (in Chapter 4), based on the employed components.

Chapter 4

RELATED WORK

4.1 Introduction

After presenting the different types of multimedia synchronization (including IDMS) in Chapter 2, and the main components of an IDMS solution in Chapter 3, this Chapter reviews the state-of-the-art in this area. First, Section 4.1 provides an overview of different works that have surveyed existing multimedia synchronization solutions, and proposed reference models to classify them. The study of the existing reference models is useful to: i) better understand this research topic; ii) identify the required features to accomplish the different synchronization demands; iii) compare the approaches and mechanisms that have been devised up to date to overcome different synchronization challenges; iv) analyze the evolution and latest advancements on multimedia synchronization; and v) identify missing as well as existing components that need further research. Likewise, a study of the overall multimedia synchronization area is a necessary task before proceeding with the design of an IDMS solution, as this type of multimedia synchronization solutions typically have to cooperate with intra-media and inter-media synchronization ones, sharing resources (e.g., feedback channel, playout buffers…) and having common components (e.g., delivery and control protocols, adjustment techniques…). After that, Section 4.2 solely focuses on IDMS, by presenting a survey of existing solutions that have been devised up to date. These IDMS solutions are then systematically classified in Section 4.3 in terms of key factors, according to many criteria and patterns from the reference models described in Section 4.1. The review and classification of existing IDMS solutions is useful to: i) confirm the relevance of specific components to enable IDMS; ii) identify the strengths and weaknesses of existing IDMS solutions; and iii) derive the (technical) requirements that must be met by the IDMS solution under design in this PhD thesis (enumerated in Chapter 6).

4.2 Multimedia Synchronization Surveys and Classification

Over the last decades, multimedia synchronization has been a live research area. Many intra-stream and inter-stream synchronization solutions have been devised since the appearance of the distributed multimedia applications. Likewise, the early development of multi-party multimedia services revealed the need for IDMS solutions, even though IDMS has recently gained relevancy due to the increasing demands for different forms of interactive shared media experiences, as previously discussed in this thesis.

The following research publications (chronologically ordered) provide a thorough review of the state-of-the art in multimedia synchronization: [Lit91], [Ehl94], [Köh94], [Bla96], [Per96], [Ish00], [Lao02], [Bor09a], [Sha12a], [Hua13], and [Men14].

In [Lit91], an analysis of the temporal and spatial composition of multimedia applications was presented. Accordingly, a classification model for both intra-stream and inter-stream synchronization, for both continuous and discrete media, was proposed. This model is composed of three synchronization levels (physical, system and human levels), although no detailed description or classification criteria are provided in this study.

In [Mey93], a taxonomy on multimedia synchronization was presented. In this study, a classification scheme was proposed, which is composed of the following three (abstraction) layers:

1) *Media layer*: it copes with intra-stream synchronization.
2) *Stream layer*: it copes with inter-stream synchronization of continuous media.
3) *Object layer*: it operates on top of the two previous layers and is responsible of offering to the multimedia application a complete and ordered multi-stream presentation, in which all types of media are correctly structured in time and space. This layer is not responsible of providing intra-stream and inter-stream synchronization, but it uses the services provided by the *Media* and *Stream layers*, respectively, for that. This layer involves the synchronization of both continuous and discrete media.

The hierarchical structure of this reference model and the abstraction level of each involved layer are shown in Figure 4.1. The services (i.e., the synchronization functionalities) provided by each layer can be accessed either directly by the multimedia application or indirectly through higher-level layers (via appropriate interfaces).

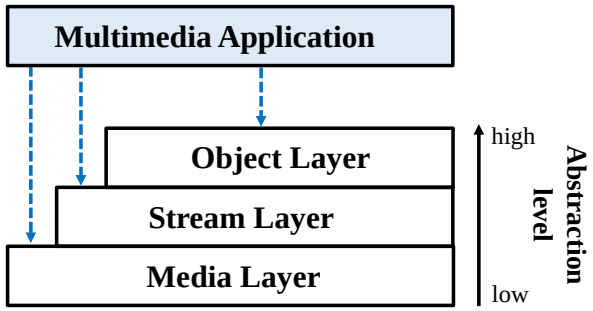


Figure 4.1. Three-Layer Reference Model in [Mey93].

In [Ehl94], a classification of existing multimedia synchronization algorithms (up to 1994) was presented. This classification was based on the location where the synchronization functionality was developed and performed: local and distributed. *Local synchronization techniques* are only implemented within workstations (i.e., media is captured/retrieved and consumed within single devices, without the intervention of networking equipment). *Distributed synchronization techniques* are used in networked environments and can also be divided into two main (sub-)approaches: i) the synchronization functionality is implemented at the sender and/or receiver devices; and ii) the synchronization functionality is implemented among the inter-network devices. Both local and distributed synchronization techniques can involve a single or multiple media sources.

In [Köh94], another reference model was presented, which makes use of three design criteria to classify the existing multimedia synchronization solutions. Each criterion is placed in different orthogonal axes, such that the overall problem space for multimedia synchronization can be graphically systematized in a 3D cube. The criteria are:

- *Time*: whether the synchronization solution makes use of global or local clocks. This criterion determines if the involved entities have an explicit common understanding of time or not. In the former case, some kind of clock synchronization takes place. Accordingly, the presentation times of MUs will be typically specified by using absolute or relative timestamps. If no clock synchronization takes place, multimedia synchronization can still be achieved by using other control mechanisms, such as markers, buffering techniques or specific audio/visual features (e.g., by means of watermarking or fingerprinting techniques).
- *Location*: whether the synchronization functionality is located at the server or at the client side.

- *Method*: the specific adjustment techniques that are performed to achieve synchronization.

In [Per96], a uniform, theoretical foundation for discussing multimedia synchronization and temporal specification was developed. A reference framework was proposed, which was used to compare existing temporal specification schemes and their relationships with multimedia synchronization.

The survey in [Bla96] summarizes multimedia synchronization requirements and proposes a reference model to compare the existing intra-stream and inter-stream synchronization methods (up to 1996). This reference model is an evolved version of the one presented in [Mey93]. In particular, it clarifies the services provided by the *Media*, *Stream* and *Object* layers and includes a fourth one, called the *Specification layer*, as can be seen in Figure 4.2. The *Specification layer* is an open layer which contains applications and tools for the creation of synchronization specifications. Examples of such tools are synchronization editors, multimedia document editors, formatting and conversion tools, authoring systems, etc. The synchronization specification will be used as an input to the *Object layer* for scheduling the overall presentation. Likewise, the synchronization specification methods can be classified into the following main categories:

- *Axis-Based specification*. This specification method aligns the MUs of each media stream by using a global (shared) timeline axis, provided either by real or virtual clock sources. For example, a real timeline axis can be provided by NTP, while a virtual timeline axis can be obtained by estimating the clock skews across distributed communication devices. This specification method allows indicating and identifying the generation and presentation instants, as well as the duration, of each MU.
- *Interval-based specification.* This specification method provides the temporal relations (i.e., the sequence order) between MUs. However, the start and finish instants of each MU, as well as their duration or the idle interval between them, is not specified. Accordingly, this specification method, by itself, cannot meet the synchronization demands of continuous media, but needs to be used in combination with other specification methods.
- *Control-Based and Event-Based specification*. In these methods, the synchronization specification is given by a set of discrete reference points, based on which the multimedia presentation can be scheduled (and re-aligned, if necessary). These reference synchronization points can be periodic or sporadic markers within the media streams (*control-based*) or even dynamically triggered events (e.g., user-generated actions or state modifications) that explicitly indicate the need of synchronization (*event-based*).

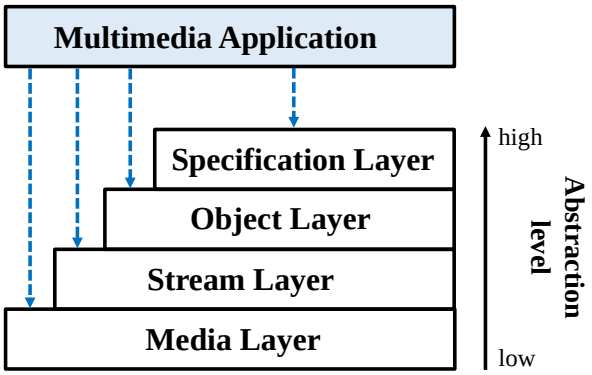


Figure 4.2. Four-Layer Reference Model in [Bla96].

Likewise, three main methods to carry out the reference synchronization information were distinguished in [Bla96]. The first one is to deliver the complete synchronization information before starting the multimedia streaming session or presentation. For instance, this is the employed method when using Nested Context Language (NCL) [NCL] and Synchronized Multimedia Integration Language (SMIL) [SMIL]. The second method consists of sending the synchronization information multiplexed within the media data streams. For instance, this is the employed method when using (variants of) MPEG technologies [MPEG]. The third one is to use a separate feedback channel to convey (additional) synchronization information. This is the employed mechanism when using RTP/RTCP protocols [Sch03].

In [Ish00], a comprehensive comparison between intra-stream and inter-stream solutions for continuous media was provided. Such solutions were compared in terms of location of the synchronization functionality, clock information, and the type of media (live or stored). Likewise, this study identified the control techniques used in each surveyed solution, classifying them into the four categories (*common control*, *basic control*, *preventive control* and *reactive control*) described in Chapter 3.

In [Lao02], a comparative survey between many intra-stream synchronization solutions (including playout adjustment techniques) is provided. The survey discusses issues related to timing information, handling of late MUs, quality evaluation metrics, and adaptation to changing delay conditions.

Likewise, several synchronization solutions for keeping consistency in Collaborative Virtual Environments (CVEs) and MOGs have been devised. Most of such solutions are event-based (e.g., [Dio99], [Pal04], [Cro04], [Mau04]), which aim to ensure that specific events are (almost) simultaneously executed by all the involved Sync Clients. In [Fle10], a synthesis of synchronization architectures and

mechanisms used by some CVEs is presented. Likewise, the works in [Roc08] and in [Fle08] provide an overview of architectures and synchronization algorithms adopted by some MOGs. In such works, the surveyed consistency maintenance algorithms are classified as either *conservative* or *optimistic* algorithms, based on how they deal with possible conflicts and the corresponding adjustments that are performed. On the one hand, *conservative algorithms* (e.g., [Dio99]) tackle the synchronization problem by preventing miss-orderings outright, allowing the processing of new events only when it is consistency-safe to do so. In *conservative algorithms*, Sync Clients are not allowed to execute new events until all the other Sync Clients have acknowledged the execution (or presentation) of the most recent one being locally executed. The occurrence of inconsistencies is impossible since no Sync Client advances its timing until it has the same exact information as the other ones. On the other hand, *optimistic algorithms* (e.g., [Pal04], [Cro04], [Mau04]) employ mechanisms to detect and correct probable conflicts, processing events optimistically before knowing for sure that no earlier events could arrive, and then repairing inconsistencies if the estimations were wrong. Optimistic algorithms are far better suited to real-time scenarios, where interactivity and responsiveness are crucial aspects.

Based on various classification criteria from [Ehl94], [Köh94], [Bla96] and [Ish00], a thorough analysis and comparison between existing inter-stream synchronization and IDMS solutions (up to 2009) is provided in [Bor09a]. This is the first survey that addresses the IDMS problem.

The study in [Sha12a] provides a classification of the existing techniques (up to 2012) to perform intra-stream synchronization, inter-stream synchronization, and IDMS. The main difference with respect [Bor09a] is the survey of intra-stream synchronization techniques.

The study in [Hua13] provides a historical review of synchronization studies for continuous media (including IDMS), by also conveying the background of technological advancements (with the associated synchronization challenges and requirements), synchronization modeling and human perceptual evaluation. That study claims an urgent need for the research community to further evolve and advance existing synchronization practices, standards and specifications. In particular, the insufficiency of the existing reference models to meet the synchronization demands in next-generation heterogeneous multimedia services was identified. For instance, telepresence systems, such as 3DTI, demand the following synchronization features: 1) heterogeneity of media modalities and devices; 2) support for scalable multi-party environments; 3) provision of the different types of synchronization; and 4) support for diverse multimedia applications on single platforms. However, the existing synchronization models mainly focus on single dimensions, such as the location where the synchronization functionality is performed [Ehl94], or the type of synchronization demands [Bla96], but do not support a combined interaction between these dimensions (i.e., the

existing models mostly cover orthogonal dimensions). As an example, Figure 4.3 illustrates the relationships (i.e., the shared layers and functionalities) and limited interactions between the reference models proposed in [Lit91], [Ehl94] and [Bla96].

Consequently, a new multi-dimensional (i.e., multi-device, multi-modal, multi-layer, multi-location and multi-activity) classification model is proposed in [Hua12] and in [Hua13]. First, this model takes into account the scalability and heterogeneity of devices and media modalities (e.g., audio, video, haptics, sensory data…). Second, this model addresses the different types of synchronization demands in four hierarchical layers, following the approach in [Bla96]. The *Media layer* has mainly the same functionality than in [Bla96]. However, the *Stream layer* is extended: i) it differentiates between inter-stream synchronization of different streams of the same media modality (e.g., arrays of video cameras or microphones) and of different modalities (e.g., lip-sync); ii) it covers both temporal and spatial correlations (e.g., location of the different sensory devices); iii) it copes with inter-sender synchronization; and iv) it copes with IDMS. The *Object* layer is not covered, since that study focuses only on continuous media, and the synchronization of discrete media is considered a solved problem with the model in [Bla96]. Likewise, the *Specification* layer is the same as in [Bla96]. Third, a multi-location dimension is added in order to encompass the end-to-end delivery chain (server, distribution and client sides), by extending the location-based model in [Ehl94]. The rationale is that the synchronization skews in a specific layer occasioned in one location (e.g., delay variability when capturing and encoding media content at the server side) are propagated to the other locations (e.g., to the network and client sides), thus having a serious impact on the synchronization performance. Therefore, a coordination between all the involved entities in the synchronization process is required. Finally, a fourth dimension is added to characterize the activity and application heterogeneity. This dimension is relevant because it is not appropriate to use a single synchronization reference model to represent all possible use cases or applications, as the requirements on temporal synchronization, and on selecting the synchronization reference (being this reference a specific device, site, stream and/or participant) are largely activity-dependent, and have a different impact on the human perception. The orthogonal dimensions, with their hierarchical structure (if any), of this synchronization reference model are illustrated in Figure 4.4.

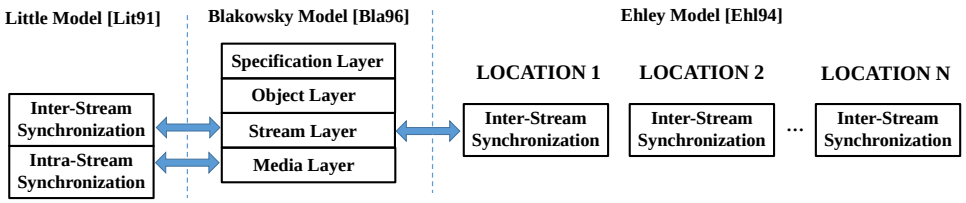


Figure 4.3. Interaction between Synchronization Reference Models.

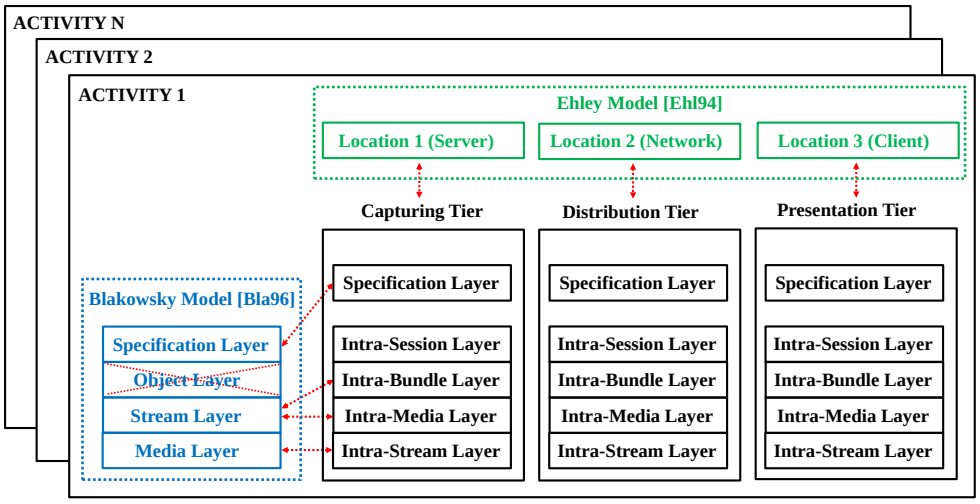


Figure 4.4. Multi-dimensional Synchronization Model in [Hua12] and in [Hua13].

The study in [Men14] provides a systematic literature review and mapping study on multiple TV content synchronization. That study surveys existing synchronization solutions, classifying them in terms of: types of involved devices, types of media content, types of synchronization techniques, targeted applications or scenarios, and evaluation methodologies. The following synchronization aspects are considered to classify the existing solutions: protocols, algorithms, delivery channels, specification methods, architectural schemes, allowable asynchrony levels and evaluation metrics. This study also claims the need for further research in this topic.

Likewise, the four-layer model in [Bla96] is slightly modified and extended in [Men14]. First, the *Specification layer* is relocated. It is argued that this layer should not be an isolated layer, but it must be bound to the other layers, since all of them need their own synchronization specification. Second, a fifth layer, called *Semantic layer*, is added on top of the *Object layer*, which has to cope with IDMS, content-based synchronization and contextual information (e.g., cross media, mash-ups…). The *Semantic* layer is responsible of communication, search, retrieval and interpretation of media content and playout timings. It has to take into account the semantic relationships between the involved types of content being consumed, as well as how to access, generate and/or consume extra related content. This layer is essential to enable advanced synchronization-sensitive services, such as personalization, interactive services, multi-screen settings, etc. In Figure 4.5, this five-layer reference model is illustrated.

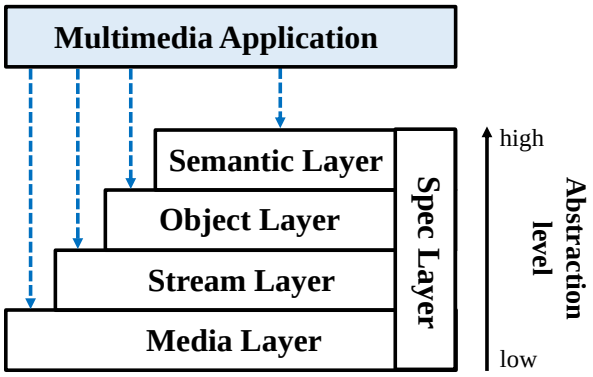


Figure 4.5. Five-Layer Reference Model (including the Semantic Layer) in [Men14].

The following example reflects the need for extending the four-layer scheme from [Bla96] to meet the requirements in interactive TV experiences. The *Media layer* is necessary to individually synchronize the involved streams (intra-stream synchronization of audio and video), while inter-stream synchronization (e.g., lip-sync) is performed at the *Stream layer*. In addition, the *Object layer* is required to correctly schedule the timing and layout of the overall presentation (probably comprising still images, graphics, advertisements, Electronic Program Guides or EPGs …). Moreover, users can require additional (live or on demand) information, related or not with the TV content being consumed. Similar issues arise when mash-ups need to be provided, when multi-screen settings need to be supported and when IDMS must be provided (e.g., in Social TV). All these functionalities will be provided by the *Semantic layer*, according to [Men14].

4.3 Brief Description of Existing IDMS Solutions

In this Section, the main features of all IDMS solutions found in literature are briefly described.

In [Esc94], an adaptive solution to achieve both inter-stream synchronization (audio and video) and IDMS over arbitrary topologies is proposed, which is called *Flow Synchronization Protocol* (FSP). FSP organizes the Sync Clients in groups, measures the delays for all flows in each group, by relying on time stamped control messages and network clock synchronization, and then enforces the largest delay for all the involved flows in that group by using buffering techniques. FSP makes use of SMS, in which a pre-designated controller (or initiator) node, i.e. a Sync Manager, is responsible of disseminating the control information to the involved Sync Clients.

In [Aky96], IDMS protocols are proposed for both real-time multimedia applications (e.g., tele-conference, tele-orchestration) and on demand services. The proposed protocols provide synchronization for both one-to-many and many-to-many configurations, and does so without prior knowledge of the end-to-end delay distribution. The only a-priori knowledge such protocols require is an upper bound for the end-to-end delay. Such protocols make use of virtual global time, instead of NTP, and perform satisfactorily even in the presence of clock drifts (i.e., when the clocks of the different users run at different rates). Such protocols adopt a DCS (but selecting a chairman as the distributor of the virtual global time), and their operation mainly consists on enforcing a global initial playout instant and then periodically enforcing playout adjustments according to delay estimations. The performance of such protocols was proved through simulation experiments in both LAN (Local Area Network) and WAN (Wide Area Network) environments.

In [Ish95], the *Virtual Time Rendering* (VTR) algorithm, which provides support for intra-stream and inter-stream synchronization, is presented. The VTR algorithm aims at adaptively keeping the temporal and causal relationships between the MUs within and among the involved streams in distributed multimedia applications. It is applicable to networks with unknown delay bounds, and consists of the dynamic adjustment of presentation times of the MUs, according to the variable network conditions, by using virtual-time contractions or expansions. In [Ish97a], the VTR algorithm was extended to also support IDMS for stored media, using an M/S Scheme, whereas it was also adapted to be used for live media in [Ish97b], by enhancing the method for estimating the network delays and by making use of a SMS. In [Ish99], the VTR algorithm was adapted to also adopt a DCS, with the goal of enhancing the fairness among the Sync Clients. In [Tas02], the influence of handover on several application-level QoS metrics (including the IDMS performance) was examined by employing VTR with SMS in an integrated wired and wireless network. In [Ish02] and [Ish03], the DCS and SMS variants of VTR algorithm, respectively, were enhanced by taking into account the importance of the media objects, for its application in networked virtual environments. In those works, the concepts of *"global importance"* (importance which is judged from the point of view of all the users) and *"local importance"* (importance which is judged from the viewpoint of each user) were introduced. In [Ish04], the SMS-based variant of VTR algorithm was enhanced to support haptic media, in addition to audio and video, with the goal of providing IDMS in networked 3D virtual environments, in which distributed users can collaboratively manipulate objects using force feedback devices. The challenge in such scenarios is that haptic media requires more stringent delay bounds than audio and video, ranging from 30 to 60 ms, according to [Sri97] and [Mat00]. In [Num05], the performance of the VTR algorithm, when using each of three IDMS control schemes (SMS, DCS and M/S Scheme), was compared in a (multicast) mobile ad-hoc network. In [Has06], the SMS-based variant of VTR algorithm was extended to be efficiently used in a networked collaborative real-time game using haptic media, such that two reference playout points (concretely, the

most advanced and the most lagged ones) could be selected as the synchronization reference. Accordingly, that study examined the influence of the determination methods of the reference playout timings on the fairness among the players (i.e., the Sync Clients) and the efficiency of the work. On the one hand, it was reported that the IDMS control improves the efficiency of the work in collaborative games if the playout timings of all the Sync Clients are adjusted to the earliest (i.e., the most advanced) one. This is because a player with earlier playout timing can help other players with later playout timings. On the other hand, it was concluded that the effectiveness of the IDMS control in competitive games, in terms of the fairness between the players, may be improved by adjusting the overall playout timing to the latest (i.e., the most lagged) one. That SMS-based variant of VTR algorithm was further enhanced in [Kur07], such that it can be efficiently applied in a P2P (Peer-To-Peer) based system for remote haptic drawing. The enhancement basically consists of implementing the functionality of the Sync Manager in each of the involved participants. In [Hos09], the DCS-variant of the VTR algorithm is also enhanced, by taking into consideration different conversation roles in a MOG (concretely in *rock, paper, and scissors* game) using a video conferencing system. Thus, the playout adjustments depended on the role (e.g., caller or receiver) of each player, similarly as in an M/S Scheme.

In [Dio99], a distributed (conservative) synchronization mechanism is proposed to guarantee the consistency of a MOG (called *MiMaze*), while preserving the real-time interactivity properties (which range from 40 ms to 200 ms for networked games, according to that work). A bucket mechanism is used for delaying the presentation of MUs a minimal amount of time such that a global consistent state of the game can be kept. This way, all the MUs generated at the same time by each of the game entities will be executed and presented together in all the involved game entities. This solution also includes a simple dead reckoning algorithm to recover from lost or late packets. This synchronization mechanism uses multicast RTP over UDP/IP as the delivery protocol, and makes use of NTP for clock synchronization.

In [Dom04], an adaptive protocol for achieving IDMS is presented, called *Multipoint Synchronization Protocol* (MSP). MSP is fully distributed (i.e., it makes use of DCS) and it is developed at the application-layer, using overlay multicast, if available. MSP was designed to be robust and to scale to large groups of Sync Clients. Likewise, MSP can synchronize concurrent media streams over best-effort packet delivery networks, in the absence of a global physical clock, and supports both turn-taking and interactive full-duplex communications.

In [Cro04], an optimistic synchronization algorithm for MOGs is proposed, called *Trailing State Synchronization* (TSS). In TSS, inconsistencies are identified by detecting when the leading state and the correct state diverge, and are corrected at that point. To recover from inconsistency situations, several instances of the game state, with different synchronization delays, are simultaneously stored. TSS makes use of DCS and of global clocks.

In [Pal04], another optimistic synchronization algorithm for MOGs is proposed, called *Interactivity-Loss Avoidance* (ILA). ILA aims at preserving the ordering and interactivity properties, while keeping a global consistent game state (IDMS). ILA makes use of DCS and global clocks. It implements a proactive event discarding mechanism, which relies on the discrimination of obsolete events. In essence, this algorithm consists of monitoring the interactivity level of the system and, when required, preempt the loss of interactivity by discarding events that have lost their importance during the game evolution. The evaluation in that work showed that ILA contributes to a better playability in a specific MOG thanks to a smoother progression of the executed events.

In [Mau04], a synchronization solution is proposed to avoid inconsistencies between Sync Clients in replicated continuous applications, such as MOGs. This solution makes use of DCS and it is mainly comprised of two algorithms. The first algorithm, called "*local lag"*, is used to compensate for short term inconsistencies. It consists of adding an extra delay to the execution timestamp of users' events, which corresponds to the largest network delay estimated for each one of the Sync Clients. This way, the "*local lag"* algorithm aims to ensure the events are received by all the Sync Clients before their execution time is reached. The value of the *local lag* needs to be small enough so that the users do not perceive the delayed execution, but large enough to ensure the events are received by all of them. A value of 150 ms for applications that span the globe is recommended in that work. However, although the *"local lag"* algorithm aims to prevent from small term inconsistencies, it does not provide any guarantees. Therefore, a second algorithm, called *"time warp"*, is proposed to undo inconsistencies that may still occur due to various uncontrollable factors. It mainly consists of rolling back changes to the last known consistent state in case that inconsistencies are detected.

In the iNEM4U (Interactive Networked Experiences in Multimedia for You)[8] platform [Hes10], the synchronization solution proposed in [Mau04], which was originally devised for networked gaming, is adapted to provide open, intelligent, and interoperable support services for social media applications. Likewise, this solution in [Mau04] is also adapted in [Vai11a] (PhD thesis) and [Vai11b] to enable coherence in shared video watching scenarios. Concretely, the *"local lag"* algorithm is adapted to provide globally synchronized playout when distributed geographically users are watching the same media content, while interacting via a chat channel (e.g., voice or text). It is achieved by sending periodic events including playout position updates (e.g., one event per 30 seconds), called synchronization control messages, and then executing these events in a timely synchronized manner at all the involved Sync Clients, by performing the required playout adjustments (if necessary). These events can also include navigation control commands, such as

[8] iNEM4U Project: www.iNem4u.eu, Reference FP7-ICT-2007-1-216647.

"pause", *"play"* or *"jump to scene"*. This way, for example, if a user pauses the media playout, then, the pause should also be executed at all the other users within bounded time limits. This synchronization algorithm was implemented using both M/S Scheme and DCS. In [Vai11a], the reference Sync Client is the one with the smallest media buffer size. The *"time warp"* algorithm was not considered in that work. Although this solution makes use of NTP for clock synchronization, an alternative ad-hoc mechanism for clock synchronization, called *neighbourCast*, is also proposed to overcome the situation in which the involved Sync Clients cannot access the same NTP server. The performance of this synchronization solution was assessed when using both DCS and M/S Scheme, obtaining indistinguishable results. Concretely, the estimated synchronization error was around 300 ± 200 ms in a WAN scenario (one user in Amsterdam and another one in Seoul) and around 150 ms (with a standard deviation of 59 ms) in a LAN scenario, with all the involved consumer devices with the same characteristics (same hardware and software configuration) and using the same NTP server. This synchronization solution is then used to study the ranges of asynchrony levels that are acceptable in distributed video watching scenarios (the findings of that study, published in [Gee11] and in [Bor13], are summarized in Section 2.8). Apart from distributed media synchronization or IDMS, the work in [Vai11a] addressed other key requirements to enable coherence in synchronous shared media experiences: QoS, time-bounded delivery and user mobility. The concept of user mobility refers to the capability of switching the media presentation between different consumption devices in a synchronized manner, with a seamless adaptation to the new session context.

The research work in [Hua12] (PhD thesis) focused on meeting the synchronization demands (including intra-stream, inter-stream, inter-sender and inter-destination media synchronization - IDMS -) in 3DTI systems. In such environments, a set of correlated multi-streams from diverse types of sensory devices are involved. These media streams are typically aggregated as a bundle at a sender gateway during transmission and then are splitted at a receiver gateway for their individual consumption. In that work, extensions to RTP/RTCP were proposed to feature new multi-modal multi-stream characteristics for scalable and interactive 3DTI systems, as well as to offer both specification and control support for realizing the demands of multi-sensory, multi-device and multi-layer media synchronization. On the one hand, RTP header extensions were proposed to include: i) 64-bit timestamps for each MU; ii) information about synchronization points across the involved streams; iii) the synchronization dependences across the involved streams (including QoS priority levels); and iv) *tob* (buffer time offset) field, which indicates the offset (i.e., the additional size) of the receiver buffer that needs to be adjusted in order to achieve synchronization. Even though the proposed solution provides support for both inter-sender synchronization and IDMS, only one of the two functionalities can be performed during the system run time. For inter-sender synchronization purposes, the *tob* field is specified by the gateway at the receiver side, while for IDMS purposes, it is specified by the gateway at the sender side. In

order to explicitly indicate whether the system provides inter-sender synchronization or IDMS, a *Type of Service* (*ToS*) field is also added to the RTP header. On the other hand, extensions to RTCP sender and receiver reports were proposed to inform about the synchronization references (both between MUs of correlated streams and between involved senders and receivers). This solution makes use of globally synchronized clocks, by means of NTP, but also relies on virtual clock synchronization to remedy clock drifts. It adopts an M/S Scheme, in which the Sync Client with the largest delay is selected as the synchronization reference. However, it is also argued that another reference Sync Client could be explicitly selected in specific activities. For instance, this would allow to select the most active participant or the leader participant (e.g., the instructor in synchronous e-learning, the director in a conferencing, or a doctor in an e-health scenario) as the synchronization reference.

In [Sha12b], a platform to provide IDMS in structured document-based media is designed and evaluated. It is implemented as a plugin for the AMBULANT Open SMIL player[9], and it basically consists of synchronizing the document clock (which is shared for all media types in a SMIL presentation) across separated Sync Clients, using both SMS and M/S Scheme. This solution does not employ globally synchronized clocks, but tries to compensate the drift between the document clocks of the Sync Clients by exchanging periodic control messages and estimating the Round Trip Time (RTT) between them. This process is called *virtual clock synchronization*. Apart from periodic exchange of timestamps (i.e., axis-based synchronization), this solution supports the synchronization of user generated events (e.g., navigation control commands) to enable coherent interactions between the involved users when consuming media together (i.e., event-based synchronization). In that work, the potential of document-based media for IDMS is discussed, since each Sync Client can select the most appropriate alternative version of the media content (e.g., resolution, format…), according to the available bandwidth and computing resources.

In [Jan13b], the IDMS-enabled multimedia engine developed in [Sha12b] is integrated with a video-mediated group conferencing testbed [Sch13] to enable shared media consumption between remote users, while interacting via audio/video conferencing. The goal is to constitute a flexible framework that allows not only having a full control about technical aspects of IDMS (e.g., clock synchronization, algorithms, adjustment techniques …), but also investigating, from the end-user's perspective, the suitability of various strategies and policies when taking into account the media content being shared, the social setting and the interaction channels in use.

[9] AMBULANT Open SMIL player, www.ambulantplayer.org.

In [Wij12a], a web-based framework, called *synchronous Media Sharing* (sMS), is proposed to enable geographically separated users to share and consume digital pictures and video clips in a synchronized fashion. Besides, this framework includes interfaces with voice conferencing tools and social networking sites to further improve the feeling of social co-presence among users. It is exclusively built around open web technologies and standards in order to maximize its deployment and market penetration potential. The sMS framework adheres to a centralized communication model, in which a server (i.e., a Sync Manager) monitors the state of the session for each client and stores the relevant information for synchronization purposes in a relational database. The synchronization messages are exchanged between the server and clients, by means of a bidirectional AJAX communication. Although it is stated that IDMS is a crucial aspect *"to create a true sense of connectedness and of concurrently and socially sharing media"*, the adopted web-based approach yields a straightforward, relatively loose synchronization, in which *"timing discrepancies amounting up to a handful of seconds might occur across participating sites"*. Authors argue that such synchronization levels may be acceptable for digital entertainment purposes and recreational applications for residential users, according to a qualitative user-centered research done in [Wij12b]. However, they recognize the limitations of the adopted approach and that stringent synchronization levels should to be provided in order to use this framework for real-time interactive scenarios, such a synchronous e-learning.

In [Rai14], a client-centric approach to provide IDMS in over-the-top (OTT) HTTP streaming using MPEG Dynamic Adaptive Streaming over HTTP (DASH) is presented. This IDMS approach adopts a distributed communication model to establish and manage different groups of clients, as well as to negotiate the IDMS reference to synchronize with. It does not imply modifications to the MPEG DASH servers. Instead, extensions to the MPDs (Media Presentation Description) files are defined to include so-called IDMS Session Objects (ISOs). The ISOs include the required information to achieve IDMS and are stored at a dedicated MPD server. Likewise, this IDMS approach integrates a novel AMP technique that considers the temporal distortion of audio and video, concretely the (visual) motion intensity and the (audio) spectral energy, to determine the most appropriate instants/periods to perform the playout adjustments, thus minimizing their impact on the QoE. This IDMS approach was evaluated with respect to scalability, traffic overhead and the time required to achieve IDMS through simulation, although the impact of the proposed AMP technique was subjectively assessed using crowdsourcing.

In [Bor08], a preliminary version of an IDMS solution, by using and extending RTP/RTCP protocols, was presented. This IDMS solution is mainly axis-based (by using the timelines provided in RTP packets), but it also relies on the exchange of regular RTCP packets to monitor and control the overall synchronization process (control-based). This solution employs an SMS to carry out the synchronization control, in which the Media Server acts as the Sync Manager, which selects a Sync

Client as the (master) synchronization reference for adjusting the playout timing of all the other (slave) Sync Clients. This IDMS solution uses NTP for clock synchronization. In [Bor09c], this RTP/RTCP-based IDMS solution was extended to be able of independently synchronizing the media playout of several logical groups (called *clusters*) of Sync Clients. The performance of such solution, in terms of inter-stream synchronization and IDMS, was evaluated, both objectively and subjectively, in real LAN and WAN scenarios, in [Bor08] and [Bor09c]. This IDMS solution was designed before the one in [Hua12] and [Hua13], which also proposes extensions to RTP/RTCP protocols, but is the last one being introduced in this Section, as it constitutes the starting point of this PhD thesis. Its main properties and limitations are described in Chapter 8.

Finally, it should be also mentioned that various multimedia applications provide some sort of IDMS in their service offering. For instance, *Yahoo! Zync*[10] is a shared video watching application integrated with *Yahoo! Messenger* (text chat tool), which allows sharing the navigation control commands (e.g., if the video playout is paused by one user, the video players of all other users are also paused). Watchitoo[11] is another web-based application that not only enables text chatting, but also audio and video conferencing, while remotely watching together the same video content. Nefsis[12] uses cloud off-loading to synchronize the playout of media files at different locations during a tele-conferencing session. One client starts to play out a file which is then sent to a virtual server in the cloud, who shares the file with all the clients in the shared session. It is not clear which technology or underlying synchronization techniques are applied in these three last applications, since they are proprietary frameworks which have not been published anywhere (and are probably under intellectual property). However, such application provide loose synchronization. They mostly focus on the synchronization of stored media files, rather than of live streams. Likewise, they mostly rely on synchronizing certain control events, such as the play/pause/stop/seek commands, but do not provide continuous synchronization.

Another example is the open-source Video LAN Client (VLC)[13] media player, which implements a functionality, called *"netsync"*, to synchronize playout at multiple player instances. It makes use of an M/S Scheme, in which the master VLC player broadcasts a clock signal to which the clocks of all the other (slave) VLC players must synchronize. However, this solution does not take into account the delay variability across Sync Clients.

[10] Yahoo! Zync: http://sandbox.yahoo.com/heres-zync Last access in December 2014.
[11] Watchitoo: http://watchitoo.com/ Last access in December 2014.
[12] Nefsis: http://www.nefsis.com/ Last access in December 2014.
[13] VLC media player: http://www.videolan.org/vlc Last access in December 2014.

4.4 Taxonomy of Existing IDMS Solutions

After briefly presenting the existing IDMS solutions, with their variants and enhancements, such solutions are classified in this section. This taxonomy of IDMS solutions updates the one presented in [Bor09a], but provides two main novelties. On the one hand, this study comprises many IDMS solutions devised since 2009 (publication date of [Bor09a]), such as [Hos09], [Bor09c], [Vai11a], [Vai11b], [Gee11], [Hua12], [Hua13], [Sha12b], [Wij12a] and [Rai14]. On the other hand, it takes into account further factors to classify the existing IDMS solutions. Concretely, the factors considered in [Bor09a] are the following (it is also indicated in parenthesis if other classification studies considered each one of these factors).

- *Control Scheme* ([Men14]): It indicates the adopted control scheme by the particular IDMS solution.
- *Clocks* ([Koh94], [Ish00]): It indicates if the IDMS solution makes use (or not) of globally synchronized clocks.
- *Network delay limits* ([Ish00]): The need for the IDMS solution to know in advance the network delay limits or their PDF.
- *MU generation periodicity* ([Ish00]): Some IDMS solutions (especially the older ones) were developed to only operate with a periodical generation of MUs, but other solutions also perform satisfactorily when the generation of MUs is non-periodic (i.e., variable).
- *Stored or live contents* ([Ish00]): Some IDMS solutions have been specifically developed for transmission of stored content, live content or for both content types.
- *Feedback Channel* ([Men14]): Some IDMS solutions use a feedback channel to exchange control messages (including information about IDMS) between the involved entities.
- *Synchronization information* ([Ish00]): It indicates the information useful for synchronization included in the (delivery packets containing) MUs or in the feedback control messages (if employed).
- *Synchronization techniques* ([Koh94], [Ish00]): It indicates the specific adjustment techniques included in each IDMS solution. The difference of this taxonomy with the one in [Bor09a] is that the location and the type of the adjustment techniques is also indicated. Such adjustment techniques were presented and classified in Section 3.6.
- *Media Synchronization Types* ([Bla96], [Ish00], [Hua13]): It indicates if the IDMS solution also provides support for, or cooperates with, other intra-stream and inter-stream synchronization solutions.

- Use of RTP/RTCP ([Hua13], [Men14]): It indicates if the IDMS solution makes use of RTP/RTCP protocols (and if such protocols are specifically used for synchronization purposes or not).

Furthermore, other relevant classification factors have been considered in this taxonomy:

- *Involved Media Types* ([Men14]): It indicates the media types (e.g., audio, video, haptic media, sensory data …) involved in each IDMS solution.
- *Application* ([Men14]): It indicates the targeted application or scenario for which the IDMS solution was designed.
- *Evaluation Metrics*: It indicates the specific metrics employed to evaluate the IDMS performance (described in Section 3.7). The survey in [Lao02] also considered this factor, but focusing only on intra-stream synchronization.
- *Evaluation Methodology* ([Men14]): It indicates if the IDMS solution was evaluated in a simulation framework or in a real environment.

Table 4.1 provides a classification of the compiled IDMS solutions, based on such factors. However, in order to not complicate this table too much and to provide a more aggregated taxonomy, the use of some of these factors has not been included in Table 4.1, but discussed separately.

The IDMS solutions have been chronologically ordered in Table 4.1, even though the different variants and evolved versions of specific IDMS solutions have been included in the same row. For example, this can be seen in the fourth row, which includes various references of IDMS solutions based on the VTR algorithm. The exception is the IDMS solution presented in [Bor08], which is the starting point of this PhD thesis, and has been included in the penultimate row, with the intention of easily checking the enhancements and extensions that have been added in the IDMS solution designed in this PhD thesis, which has been included in the last row.

The name of each particular IDMS solution (if assigned by their authors) and the associated references are included in the first column. Likewise, the gaps in the cells reflect that the related factor (column) is not considered by the IDMS solution (row) or, possibly, that no mention about the use or inclusion of that factor has been found in the compiled references.

It can be seen from Table 4.1 that each IDMS solution presents a different combination regarding the analyzed factors.

All the surveyed IDMS solutions also include intra-stream and inter-stream synchronization mechanisms (most of the later ones described in [Bor09a]). Likewise, most of them rely on global timing mechanisms.

As in [Bor09a], the IDMS solutions that make use of RTP streaming have also been identified, being the following ones: [Dio99], [Bor08], [Bor09c] and [Hua12]. Moreover, other synchronization solutions have also made use of RTP streaming (although such protocols are not used to provide IDMS), such as [Tas98], [Tas00], [Kuo01], [Cha07], [Ler07], [Lu09a], [Mag09], [Mar09], [Yun10], [Bel10], [Aga11], [How11], [Bel12a], [Bel12b], and [Riv13]. This reflects the relevance of RTP/RTCP protocols for enabling synchronization-sensitive media services. The other compiled IDMS solutions define new proprietary and ad-hoc (i.e., application-specific) protocols, with their specific metadata and control messages (using or not a feedback channel, as reflected in Table 4.1) to achieve the IDMS goal. This makes inter-operability between implementations and domains very difficult.

Table 4.1. Classification of IDMS Solutions

IDMS Solution	Control Scheme	Clock	Delay Limits	MU Generation Rate	Content	Feedback Channel	Sync Information	(Type/Location of) [14] Synchronization Techniques	Evaluation Metrics
Flow Synchronization Protocol (FSP) [Esc94]	SMS	Global	Unknown	Variable	Stored + Live	Yes	Timestamps, Flow and Group Identifier	- (P/S) Initial transmission and playout instant. - (R/C) Playout duration extension and reductions	- Tolerable asynchrony levels
[Aky96]	DCS	Local	Known	-	-	-	Timestamp in 1st packet	- (P/S) Initial transmission and playout instant. - (R/S+C) Master/Slave receiver switching (chairman). - (CM/C) Playout rate adjustments (receiver's clock).	- Maximum, minimum and mean value of the asynchrony - Number of discontinuities and Total Discontinuity Time.
Virtual Time Rendering (VTR) ([Ish97], [Ish97b], [Ish99], [Ish02], [Tas02], [Ish03], [Ish04], [Num05], [Has06], [Kur07], [Hos09])	M/S Scheme ([Ish97])	Local	Unknown	Variable	Stored	Yes	Timestamps, and Sequence numbers	Adjustment techniques used in different versions of the VTR-based IDMS solutions: - (P/S) Interleaving MUs. - (P/S) Initial transmission instant. - (P/C) Preventive pauses. - (P+R/C) Change of the buffering time. - (R/S) Decreasing the number of media streams. - (R/C) Reactive skips and pauses. - (R/C) Playout duration extensions and/or reductions - (R/C) Virtual local time expansions and/or contractions. - (CM/S) Skips at the Server side. - (CM/S) Media Scaling. - (CM/C) Playout rate adjustments.	- AvgP - MdP - Average MU delay. - MSE_{IDMS}. - Coefficient of variation of Output Interval - MU discard rate. - Total Pause Time. - Synchronization Delay. - Inconsistency Rate. - Reversal Rate. - MOS.
	DCS ([Ish99], [Ish02], [Hos09])	Local			Stored + Live	No[15]			
	SMS ([Ish97b], [Tas02] , [Ish03])	Global			Live	Yes			

[14] See the nomenclature used in Table 3.3.
[15] But timing information is exchanged between Sync Clients.

	Enhanced SMS ([Ish04], [Has06], [Kur07])	Global			Stored + Live	Yes		- VTR Techniques (upper cell). - (R/C) Event-based synchronization control. - Combination of the selection of two reference output timings (the most advanced/lagged playout points).	
Bucket Synchronization [Dio99]	DCS	Global	Known	Variable	Live	No	Timestamps, and Sequence numbers	- (R/C) Skips (discardings) and pauses (duplicates). - (R/C) Late events are dropped.	- Average MU delay. - Asynchrony Evolution. - Packet loss.
Local Lag and Time Warp [Mau04] and evolved versions ([Hes10], [Gee11], [Vai11a], [Vai11b])	DCS (Also M/S Scheme in [Vai11a])	Global	Unknown	Variable	Stored	No	Timestamps	- (R/C) Event-based synchronization control. - (R/C) Rollback based techniques. - (R/C) Playout duration extension.	- Asynchrony Evolution. - Computational Load. - Subjective evaluation (Degradation Category Rating or DCR MOS) in [Vai11a]
Trailing State Synchronization (TSS) [Cro04]	DCS	Global	Unknown	Variable	Stored	No	Timestamps		- MU discard/loss rate. - Number and magnitude of Rollbacks. - Synchronization Delay.
Interactivity Loss Avoidance (ILA) [Pal04]	DCS	Global	Unknown	Variable	Stored + Live	No	Timestamps	- (P/C) Preventive MU/event discarding. - (R/C) Event-based synchronization control. - (R/C) Playout duration extensions. - (R/C) Reactive events discarding.	- MU discard/loss rate. - Synchronization Delay.
Multi-Point Synchronization Protocol (MSP) [Dom04]	DCS[16]	Global (Virtual Clock)	Unknown	Constant	-	Yes	Timestamps, Group Identifier	- (P/S) Initial transmission and playout instant. - (CM/C) Playout rate adjustments (receiver's clock).	- Traffic Overhead.

[16] From the involved entities, one of them is dynamically selected as a controller, being responsible of the initialization and control of the protocol and of establishing the virtual global time.

[Hua12], [Hua13]	M/S Scheme	Global	Unknown	Variable	Live	Yes	Timestamps, sync Points across streams, media sync types, buffer offset	- (R/C) Increase Buffering Delays	- End-to-end Delay - Computational load - PESQ [ITU-T P.862] - Multi-view video frame rate - Asynchrony - Response Delay - Subjective Evaluation (Comparative MOS, CMOS)
SMIL plugin for IDMS [Sha12b]	SMS and M/S Scheme	Local (Virtual Clock Sync)	Unknown	-	Stored	Yes	Timestamps and navigation control commands	- (CM/C) Adjustment of the clock rate (speed up or slow down).	- Asynchrony evolution. - Maximum, minimum and mean value of the asynchrony. - Moving Average of the asynchrony evolution.
Self-Organized IDMS for DASH [Rai14]	DCS	Global	-	Variable	Stored		Timestamps, Sequence numbers, and Group Identifiers	- (C/S) Master Reference Selection Policies. - (C/S) Independent Synchronization for Differents Logical Groups. - (R+P/C) Smooth Playout rate adjustments (AMP).	- Scalability. - Traffic Overhead. - Synchronization Delay - Subjective evaluation (for AMP).
Preliminary Version of RTP/RTCP-based IDMS Solution [Bor08], [Bor09c]	SMS	Global	Unknown	Constant	Stored + Live	Yes	Timestamps, Sequence numbers, and Source and Group Identifier.	- (P/S) Initial playout instant. - (R/S) M/S switching. - (R/C) Reactive skips and/or pauses at the client side. - (C/S) Independent Synchronization for Differents Logical Groups	- Total Paused MUs. - Total Skipped MUs. - CDF of asynchrony. - Playout Delay Evolution. - Traffic Overhead. - Subjective evaluation (MOS)
Newly Designed RTP/RTCP-based IDMS Solution	SMS, DCS, and M/S Scheme	Global	Unknown	Variable	Stored + Live	Yes	Timestamps, Sequence numbers, Source Identifier, and Group Identifier.	- Techniques in the upper cell. - (P/C) Control of Buffer Fullness Level. - (R+P/C) Smooth Playout rate adjustments (AMP). - (C/S) Dynamic Master Reference Selection Policies	- Metrics in the upper cell. - Buffer fullness variation. - Asynchrony evolution. - Computational Load. - Maximum, minimum and mean value of the asynchrony. - Playout rate variation values.

The existing IDMS solutions were designed for specific applications, such as audio/video streaming (e.g., [Esc94], [Ish97a], [Dom04], [Vai11b], [Rai14]), conferencing (e.g., [Ish97b], [Has06], [Kur07], [Hos09]), MOG (e.g., [Dio99], [Ish02], [Cro04], [Pal04], [Mau04], [Roc08]), CVE (e.g., [Ish03], [Ish04], [Has06], [Fle10]), synchronous e-learning (e.g., [Ish97b], [Kur07]), and 3DTI ([Hua12]). However, some of them can be applied in diverse use cases, such as the ones based on VTR algorithm, [Dom04], [Mau04], [Bor08] and [Hua12]. The involved media types in each IDMS solution are also miscellaneous, such as audio and video (all of them), haptic media (e.g., [Ish04], [Has06], [Kur07]), user-generated events (e.g., [Dio99], [Mau04], [Vai11b], [Sha12b]), text chat ([Vai11b], [Gee11]), or sensory data ([Hua12]).

The analyzed IDMS solutions also significantly differ in the type and location of the employed adjustment techniques. This is mainly because the applicability of specific adjustment techniques is highly dependent on the involved media types, on the specific encoding mechanisms being used, on the real-time properties of media content being transmitted (live or stored), and on the particular application for which the IDMS solution has been designed.

Likewise, the IDMS solutions have adopted different architectural schemes. This is because such solutions were designed for being implemented in specific networked environments and for meeting specific requirements. As this is a key aspect for IDMS, the feasibility and suitability of the different control schemes for IDMS is exhaustively analyzed in this PhD thesis, both in a qualitative manner (in Chapter 5) and through simulation tests (in Chapter 11).

Finally, the surveyed IDMS solutions also differ in the employed evaluation methodology. Most of the IDMS solutions have been evaluated in real networked environments, but simulation techniques have also been used in [Aky96], [Num05] and [Rai14], even the IDMS solution proposed in [Dom04] was not evaluated. In some works, network emulators, such as *NIST netem* [NIST] (in [Has06] and [Hos09]), and data link simulators (in [Ish02]), are used to force specific networking conditions. In this context, the employed metrics to evaluate the IDMS performance also differ in the analyzed works, as they are also dependent on the involved media types and targeted use cases. No standard evaluation methodologies or metrics are available to evaluate, both objectively and subjectively, the IDMS performance. This is a general problem in media synchronization, not only in IDMS.

Summarizing, the compiled IDMS solutions use a large variety of options regarding the analyzed factors (even from different categories), they have been mainly devised for specific applications, with different technical requirements, and have been implemented in different networked environments, with different available resources. Besides, there is no common benchmark and the employed evaluation methodology is not consistent in the analyzed works. A further issue is that, in many cases, the papers in which the IDMS solutions are presented do not

provide enough documentation, and some important details are missing. Therefore, it is not easy to fully specify the relationships between the existing IDMS solutions and to compare them, even qualitatively.

The assessment of a quantitative comparison among the different IDMS solutions in order to clarify which one performs best becomes even more complicated. This would require to implement them in the same application, and to evaluate their performance in the same scenario under exactly the same conditions, which is extremely difficult and time-consuming. Also, subjective assessments should be conducted to analyze the user satisfaction (QoE) in each one of the IDMS solutions under comparison.

However, the goal of this taxonomy has not been to rank the existing IDMS solutions from best to worst, but to qualitatively classify them according to the analyzed factors, to check the relevance of such factors for IDMS, and to examine the specific design criteria that have been adopted in each IDMS solution. This is very useful to identify the most common (and, ideally, the most convenient) approaches for IDMS as well as to understand how the IDMS solutions have evolved along the years.

4.5 Summary

Several research works have addressed the IDMS problem up to date, including technical papers proposing and evaluating potential solutions, surveys, and two recent PhD theses ([Vai11a] and [Hua12]).

In this Chapter, the existing classification models for multimedia synchronization have been described, identifying the different layers and dimensions (either aligned or orthogonal) such models cover, as well as their interactions and shared components (if any). It is beyond doubt that the availability of reference models for multimedia synchronization aids in understanding this broad research topic. First, it facilitates the use of a common vocabulary when discussing synchronization aspects. Second, it allows to systematically integrate and classify the existing alternatives required to fulfil the overall synchronization demands.

Most of the proposed models cover the complex multimedia synchronization problem in hierarchical layers, with different levels of abstraction. A layered solution helps in reducing the complexity, by sub-dividing responsibilities, and also allows the re-utilization of specific layers in different settings. Clear examples of the convenience and relevance of layered models can be found in the computer communications world, both through the Open System Interconnection (OSI) and Transmission Control Protocol / Internet Protocol (TCP/IP) protocol stacks. However, the lack of rigorous and consistent layering policies in existing reference models has been identified in this Chapter. On the one hand, clearer specifications of the synchronization services provided by each layer is necessary. On the other

hand, the relationships and interactions between the involved layers are not sufficiently detailed.

An evolution (not revolution) of the subsequent proposed reference models has been noticed. The most recent ones have proposed modifications, enhancements and extensions to the older ones, trying to support the extra synchronization demands that emerging multimedia systems have imposed. This in some way reflects the coherent understanding of this research area and the certain convergence of the approaches that have been devised to overcome the emerging synchronization challenges.

Acknowledging the advancements in this area, at present there is no a reference model that efficiently encompasses the overall problem space for multimedia synchronization (which was represented in Figure 2.4). Nonetheless, the proposed reference models in [Hua13] and [Men14] have represented a significant progress. On the one hand, the model in [Hua13] takes into account several relevant dimensions. On the other hand, the model in [Men14] introduces the *Semantic layer*, which is a good point. However, the IDMS functionality should not be located at this layer, as recommended in [Men14]. To our understanding, the *Semantic layer* not only has an impact on IDMS, but also on the different types of multimedia synchronization techniques. For instance, the *Semantic layer* is essential to enable the dynamic triggering of content-based synchronization, which is a key feature for the different types of multimedia synchronization techniques, not only for IDMS. Therefore, the IDMS functionality should be located at a different layer, which interaction capabilities with the *Semantic* and *Specification layers*. The model in [Hua13] proposes to integrate the IDMS functionality at the *Stream layer*. Although it seems a more appropriate location than the *Semantic layer*, it would require the division of the *Stream layer* into sub-layers to properly support the different synchronization services: inter-stream (including inter-sender) synchronization and IDMS. These sub-layers, although being independent, would have to be able to perform simultaneously and also cooperate between them, if necessary (the synchronization solution in [Hua12] provides inter-sender synchronization and IDMS, but only one of both can be performed during running time).

A key missing component in the existing reference models is the support of user-level synchronization. The users are the most important "components" (i.e., the mainstay) of multimedia systems and, therefore, synchronization solutions need to take into account the users' needs, interests, interactions, as well as perceptual and contextual issues. This necessity has been to some extent identified in previous works. In [Lit91], the need of supporting the users' interaction in the (temporal and spatial) multimedia composition processes was identified (but neither details nor solutions were provided regarding such issue). Besides, the model in [Hua13] considers the impact of the activity or application heterogeneity on the *"human perception and interests"*. In addition, the work in [Jan13b] also identified the need

of *"user-level"* synchronization. However, such a need is still not sufficiently reflected in current reference models, and further work on this matter is necessary.

Finally, in the second part of this Chapter, the most thorough, complete and updated review and taxonomy of existing IDMS solutions has been provided. This review has helped to better understand the components and mechanisms required for IDMS, the strengths and weaknesses of existing IDMS solutions, and to derive the technical requirements that must be provided in the IDMS solution under design in this PhD thesis (listed in Chapter 6).

Chapter 5

QUALITATIVE COMPARISON AMONG ARCHITECTURAL APPROACHES FOR IDMS

5.1 Introduction

In Chapter 3, the different architectural approaches and control schemes for IDMS were identified. Likewise, the control scheme in use was one of the classification factors in the taxonomy of IDMS solutions presented in Chapter 4. This Chapter provides a discussion about the suitability and applicability of both the architectural approaches and control schemes for IDMS. First, some advantages and disadvantages of network-based IDMS approaches compared to terminal-based IDMS approaches are briefly discussed in Section 5.2, mainly focusing on an IPTV context. Then, a thorough qualitative comparison among the control schemes for IDMS is provided in Section 5.3.

5.2 Network-based vs Terminal-based Approaches

Although this PhD thesis is mainly focused on terminal-based IDMS approaches (as discussed in Section 3.4), this Section briefly describes the advantages and disadvantages of network-based IDMS approaches, based on the conclusions in [Sto10].

On the one hand, the main advantages of adopting network-based IDMS approaches are:

- *Scalability*. Network-based approaches can scale very well. As many end-user's terminals (User Equipment or UE) can be synchronized by a single Sync Client (e.g., located at an edge node), the number of synchronization messages will be

significantly lower than in terminal-based IDMS approaches. This will also limit the needed capacity at the Sync Manager, although at the cost of extra functionality on the edge nodes. Likewise, the synchronization buffer for media streams is shared by many UEs.

- *UEs complexity*. In network-based approaches, the UEs do not have to implement any IDMS functionality. Therefore, legacy reception devices can be employed. As an example, IPTV companies can provide to their customers a (free) STB, which would save the costs for those STBs, but at the cost of extra functionality in the network.

- *Synchronization control*. In network-based approaches, the Sync Clients are under complete control of the network provider (e.g., IPTV provider), which can guarantee the synchronization of streams sent to the UEs. If the Sync Clients are implemented at the edge of the network, small or no delay differences will occur between UEs (if their playout buffering settings are identical or similar).

- *Delay*. In network-based approaches, the buffering is performed in the network (e.g., at an edge node). Therefore, if all broadcast channels are being buffered for a short period of time at the edge nodes, the use of a network-based IDMS approach can result in shorter channel change (i.e., zapping) delays compared to when using terminal-based approaches. However, this may imply that the new channel will be delayed for certain UEs compared with other UEs not participating in the shared media experience.

On the other hand, network-based IDMS approaches also present some disadvantages. They will not work for OTT IPTV services, since network control is required, and experience has demonstrated that network providers are not eager to open their networks in this manner. Beside, network-based IDMS approaches are much more difficult to deploy in such cases in which the geographically distributed users are divided into different logical groups, which must be independently synchronized. This is because the same media stream should be delayed differently for each of the involved groups. Moreover, any delay differences between the Sync Clients and the UEs cannot be compensated (if no additional synchronization mechanisms are implemented in the UEs).

Summarizing, the main advantage of terminal-based IDMS approaches is that they do not require any changes to the network, while the main advantage of network-based IDMS approaches is that they do not require any changes to the UEs. Hence, both approaches have different rationales and impacts on the architecture of the CDN. While terminal-based IDMS approaches require updates to existing reference points and corresponding protocols, network-based IDMS approaches require new functional entities and new associated reference points [Sto10]. Likewise, network-based IDMS approaches are better suited for large-scale synchronization of commodity services, while terminal-based IDMS approaches are more cost-effective for services involving (perhaps many) small groups of users.

5.3 Comparison among IDMS Schemes

This Section presents a thorough qualitative comparison among the IDMS control schemes (SMS, M/S Scheme and DCS) described in Chapter 3. The goal of this comparison is to reveal their effectiveness and suitability for specific networked environments and applications' requirements. The following key aspects for IDMS have been taken into account in this comparison: *robustness, scalability, traffic overhead, flexibility, location of control nodes, interactivity, consistency, causality, fairness, coherence,* and *security*. The findings of this comparison are based, partially, on conclusions of previous related works (e.g., [Dio99], [Ish03a], [Nun05], [Vai11a], [Sha12b] and [Hua13]) and on our research on this topic last years.

1. *Robustness*. This refers to the ability to perform the IDMS control despite disconnections and failures of Sync Clients. In general, centralized schemes are less robust than distributed schemes and this is also the case here. In centralized schemes, if the Sync Manager (in SMS) or the master Sync Client (in M/S Scheme) fails to communicate with the Sync Clients owing to some trouble (e.g., due to congestion or firewall blocking issues) [17], the latter cannot perform IDMS control, and therefore will lose synchronization. Nonetheless, the failure of any of the Sync Clients in a distributed architecture (DCS) has a minor effect on the other Sync Clients, because each one of them has locally all the necessary information for synchronizing at any time. Hence, a server-less architecture can greatly simplify the deployment and maintenance of synchronization-sensitive distributed applications.

   Another issue is that in shared media experiences the Sync Clients are frequently joining and leaving the session. When using M/S Scheme, if the master Sync Client suddenly leaves the session without announcement, the IDMS control will fail immediately. Therefore, dynamic master Sync Client re-election policies would be needed. This is not an issue when using SMS and DCS, since another master Sync Client can be selected from the other received IDMS reports

2. *Scalability*. This refers to the ability of concurrently handling the playout timings of multiple Sync Clients in an IDMS-enabled session. SMS may present higher scalability constraints because it requires the maintenance of a dedicated Sync Manager to which all the control information converges (i.e., the Sync Manager must gather the IDMS reports from all the Sync Clients). Using SMS, if the IDMS reports are sent by the Sync Clients at a non-adaptive rate (e.g., after the playout times of specific MUs), multiple IDMS reports may be received almost simultaneously by the Sync Manager, thus

[17] It should be noted that we are assuming in this study that the Sync Clients can stablish communication with the Sync Manager (in SMS) and with the other Sync Clients (in DCS and in M/S Scheme), which may not be the case in cross-domain applications.

originating a feedback-implosion problem. Consequently, as the number of Sync Clients increases, bursty traffic due to IDMS reports can overwhelm the Sync Manager. This could also degrade the playout quality of the media streams, as the increase of network traffic could originate losses of control and data packets. This failure issue also applies to DCS, as each distributed Sync Client has to gather the IDMS reports from all the other Sync Clients (full-mesh communication process). However, the computational resources become overloaded later at a larger group size using DCS compared with using SMS. This is especially relevant when the Sync Clients are divided into different logical groups, which can be synchronized separately. The reason is because in DCS the Sync Clients only have to process the IDMS reports from the other Sync Clients belonging to the same group with whom they are sharing a media experience. In SMS, however, the Sync Manager must still process the IDMS reports from all the groups in the session (although it may also facilitate the IDMS control, e.g. comparison of the playout processes only within each group).

3. *Traffic Overhead.* This factor is closely related to the previous one. Regarding traffic overhead, two issues can be differentiated. The first one is the distribution of IDMS reports from the Sync Clients to the Sync Manager in SMS, or between the Sync Clients in DCS and M/S Schemes. In M/S Scheme, only the master Sync Client sends IDMS reports (typically in a multicast way) to the slave Sync Clients. Therefore, the network load will not be significantly increased when including IDMS control. In SMS, all the Sync Clients send IDMS reports (typically in a unicast way) to the Sync Manager. In DCS, all the Sync Clients exchange IDMS reports (typically in a multicast way). Therefore, the traffic overhead will be higher in DCS than in SMS, and higher in SMS than in M/S Scheme. The second issue is related to the transmission of IDMS setting instructions. Unlike in DCS and M/S Scheme, in which distributed Sync Clients can directly adjust their playout timing according to the incoming IDMS reports from other the Sync Clients, in SMS, the Sync Manager must send an additional control message to the Sync Clients every time an asynchrony situation is detected, which slightly increases the network load. Generally, even considering this, the traffic overhead may be higher in DCS than in SMS.

4. *Interactivity.* The lowest delays may be achieved using M/S Scheme because, unlike the other two schemes, each slave Sync Client can compute the asynchrony every time it receives an IDMS report from the master Sync Client. Delays in DCS are a bit larger because in that case each Sync Client must gather the IDMS reports from all the other active Sync Clients (probably sent and received at different instants). Then, the highest delays occur when using SMS because a bidirectional communication is required in such a case (first: the Sync Manager must collect all the IDMS reports from the Sync

Clients; second: the Sync Manager may have to adhere to some specific timing rules, having to wait before being able to send a new control message including IDMS setting instructions; and third, that control message has to be received by all the Sync Clients). Therefore, asynchrony situations will be detected and corrected earlier using M/S Scheme than using DCS, and earlier using DCS than using the SMS.

As discussed, the interval between consecutive IDMS reports should be dynamically adjusted (scaled up) if the number of distributed Sync Clients significantly increases. However, the lower interval for sending IDMS reports, the sooner the playout timing information from the distributed Sync Clients will be available. It would obviously affect the interactivity and the frequency at which the IDMS control can be performed. Consequently, the most (less) affected scheme would be DCS (M/S Scheme), because in such a case the amount of exchanged control traffic regarding IDMS is the highest (lowest) between the considered schemes.

5. *Location of the control nodes.* Centralized schemes are more sensitive to the location of the sync entities [Ish03a]. Under heavily loaded network conditions, the IDMS performance using SMS can be lower compared to the one using the other two schemes if the Sync Manager is collocated with the Media Server. This is due to the fact that the IDMS control messages sent by the Sync Manager are (or could be) sent through the same path as that of MUs (encapsulated in data packets). Although IDMS control messages may hardly increase the network load, such extra traffic could cause that some data or control packets may be dropped when the bandwidth availability is scarce. If an IDMS control message is lost, the Sync Client cannot determine the reference playout timing to synchronize with until the reception of the next IDMS control message. Conversely, in M/S Scheme, if the most heavily loaded Sync Client is selected as the master, the data packets are less likely dropped on the intermediate links, as it does not need to receive IDMS reports and its own sent ones may be transmitted in the opposite path to the one followed by MUs. Therefore, in congestion situations, M/S Scheme may achieve better IDMS performance than SMS and DCS. However, the most heavily loaded destination cannot always be known and, therefore, the master Sync Client could not be selected accordingly.
6. *Consistency.* In media sharing applications, consistency is required to guarantee concurrently synchronized playout states in all the distributed participants. SMS is commonly used in distributed games to maintain a worldwide view of the game, as a single server simplifies problems related to causality and replication consistency [Vai11a]. In centralized schemes, inconsistency between Sync Clients' states occurs less likely, since all of them always receive the same information about IDMS timing from the Sync Manager (in SMS) or from the master Sync Client (in M/S scheme).

Contrarily, in DCS there is no guarantee that the same reference IDMS timing, from among all the collected IDMS reports, will be selected in all the distributed Sync Clients, since each one takes its own decisions locally, leading to a more probable potential inconsistency situations. Also, if the IDMS reports are sent using a non-reliable transport protocol, such as User Datagram Protocol (UDP), some Sync Clients may and some other Sync Clients may not receive certain IDMS reports. This may lead to even more potential inconsistency in DCS.

7. *Coherence.* This concept refers to the ability of simultaneously coordinating the media playout timing according to a common reference timing for IDMS. In DCS and in SMS, the maximum playout asynchrony (the one between the most lagged and the most advanced Sync Clients) can be estimated. However, in M/S Scheme, slave Sync Clients can only know the asynchrony between its local playout process and that of the master Sync Client. Therefore, using M/S scheme, the reactive synchronization actions will not be performed simultaneously, because slave Sync Clients will adjust their playout timing every time they detect an asynchrony situation with the master Sync Client, and this situation may not be detected at the same time in all the slave Sync Clients. As a result, despite the fact that SMS and M/S Schemes are the most appropriate in terms of consistency, SMS outperforms the other schemes (M/S Scheme and DCS) in terms of coherence. So, it can be concluded that SMS is the best ranked scheme for IDMS regarding such factors.

8. *Causality.* Causality in multimedia synchronization refers to the maintenance of the correct chronological order of specific events. Therefore, causality control is required by interactive media sharing applications for preserving and/or restoring the original media timing. The study in [Ish03a] concluded that SMS is slightly superior to DCS in terms of causality and intra-media synchronization quality, mainly due to the minor traffic overhead. Similarly, it can be deduced that when using M/S Scheme the performance in terms of causality is superior compared to when using the other IDMS schemes, due to the same reason.

9. *Flexibility.* Using M/S Scheme, there is no option for selecting the reference (playout) timing for IDMS, since it is specified in the IDMS reports from the master Sync Client. Conversely, the Sync Manager in SMS, and the Sync Clients in DCS, can employ several dynamic policies for selecting the reference timing from all the collected IDMS reports (possible strategies will be presented in Section 8.8). Furthermore, if the Sync Clients are divided into independent logical groups, when using DCS, each Sync Client has only to register those IDMS reports from the other Sync Clients belonging to the same group, despite it may receive the IDMS reports from all the Sync Clients in the session (if sent to a common multicast address). Therefore, DCS outperforms the other IDMS schemes in terms of flexibility.

10. *Fairness.* M/S Scheme is suitable for applications in which a single Sync Client has a certain priority level over the other Sync Clients. For example, in multi-party multimedia conferencing (e.g., synchronous e-learning), the chairperson's (e.g., the teacher's) terminal can be selected as the master Sync Client. However, this scheme cannot treat all the Sync Clients fairly. This problem is minimized when SMS and DCS are employed, because the reference playout timing for IDMS is selected after a comparison among the IDMS reports sent by all the Sync Clients. As an example, the study in [Ish03a] concluded that the effectiveness of the IDMS control in competitive networked environments, in terms of fairness between the Sync Clients, could be improved by adjusting the playout timings of all the Sync Clients to the latest (i.e., the slowest or most lagged) one. Likewise, DCS may outperform SMS in terms of fairness because asynchrony situations can be corrected earlier, due to the minor network and processing delays. However, for that purpose, all the distributed Sync Clients should be coordinated to select the same reference playout point for IDMS.

11. *Security.* Another advantage of centralized architectures is that the presence of a server makes cheating difficult. In a completely distributed architecture (DCS), each Sync Client takes its own decisions, resulting in a lack of control of what each one is doing and whether a Sync Client is malicious or not. In M/S Scheme, this problem can be minimized if the IDMS process of the master Sync Client is under control. In SMS, the Sync Manager can use some mechanisms to check the validity of the arriving IDMS reports and guarantee an overall synchronization status. Hence, cheating is more difficult in centralized schemes (SMS and M/S Scheme) than in distributed ones (DCS).

    In each one of the considered IDMS schemes, the reporting of an erroneous playout point, either accidental or malicious, may lead to undesired behavior. According to the adopted model, extremely advanced/delayed playout information (e.g., several seconds) would produce large adjustments of the receivers' playout processes with the consequent loss of real-time or continuity perception. It would obviously affect the consistency, fairness and real-time interaction of the multimedia service. Therefore, the involved sync entities (Sync Manager in SMS, or each Sync Client in DCS and in M/S Scheme) should consider inconsistent playout information, exceeding configured limits, as a malfunction service and reject that information in the calculation of the necessary playout adjustments (even though that information comes from the master Sync Client in M/S Scheme).

From the above comparison, it is confirmed that each one of the IDMS control schemes has its own strengths and weaknesses. As a summary, Table 5.1 includes a ranked comparison among these control schemes regarding each analyzed factor. It is important to note that this is not an arithmetic weighting, but the numbers 1-3 are employed to classify their appropriateness regarding each factor. The weight of the

relative importance of each of these factors will depend on the specific context and space in which a specific IDMS solution is going to be deployed, as well as on the (technical) requirements that must be accomplished. Depending on the targeted goals and on the available resources, an implementer or application developer can choose to give more preference to interactivity than to traffic overhead, or more to flexibility and robustness than to security, or more to coherence than to scalability, etc. Also, such decisions can vary depending on the specific situation in which the same type of media sharing application is going to be deployed. For instance, some applications can be small-scale, while others can be deployed over large-scale settings. Some application may require the achievement of stringent synchronization levels, while lower synchronization accuracy may be acceptable in other ones. Bandwidth availability and multicast feedback capabilities can be an issue (or not) in specific scenarios. Likewise, other aspects, such as delay minimization or robustness, can be especially relevant in particular environments. At the end, the targeted use cases dictate the requirements, which will determine the necessary characteristics and functionalities of the IDMS solution under design.

Table 5.1. Qualitative Comparison among Control Schemes for IDMS

		Factors									
		Robustness	**Flexibility**	**Traffic Overhead**	**Scalability**	**Interactivity**	**Fairness**	**Consistency**	**Coherence**	**Causality**	**Security**
Scheme	**DCS**	1	1	3	2	2	1	3	2	3	3
	SMS	2	2	2	3	3	2	1	1	2	1
	M/S	3	3	1	1	1	3	2	3	1	2

1. Best Scheme, 2. Good Scheme, 3. Worst Scheme

Moreover, the differences between the suitability of the considered IDMS schemes regarding all the analyzed factors are not uniform. This means that a specific IDMS scheme can be the worst regarding a given factor (for instance, SMS regarding scalability), but this fact does not have to imply a serious constraint, only that the other schemes perform better regarding that factor. Similarly, the qualitative differences (i.e., most suited, medium suited and less suited) between the IDMS schemes do not have to be necessarily uniform for each analyzed factor. For instance, those differences may be significant regarding specific aspects (e.g., robustness, traffic overhead or interactivity), but they may not be important regarding other aspects (e.g., causality or scalability). Therefore, no definitive rules can be given, but only indicative guidelines that can be followed in the design of an IDMS solution.

With the above issues in mind, it can be appreciated from Table 5.1 that M/S Scheme can provide the best performance in terms of scalability, traffic overhead, interactivity (low delays) and causality, but presents serious drawbacks if some features such as robustness, coherence, flexibility and fairness must be provided. Therefore, M/S Scheme can be suitable in those scenarios in which the bandwidth availability is scarce (since only the master Sync Client sends IDMS reports), and also in those use cases in which a single participant (the master) has a certain priority level over the others, as in synchronous e-learning scenarios (in which the terminal of the instructor should be selected as the master reference for IDMS).

DCS is a suited option for IDMS in those use cases in which high performance in terms of robustness, fairness, flexibility, scalability and interactivity is desirable, despite of a slight cost in terms of traffic overhead (because all the Sync Clients send IDMS reports in a multicast way) [18], consistency or security. Therefore, it can be concluded that DCS can be an appropriate solution for controlled environments in which bandwidth availability is not a problem, and security aspects can be ensured, as these are the main weaknesses of DCS for IDMS (see Table 5.1).

Using DCS, the distributed Sync Clients have to process the incoming IDMS reports from all the other Sync Clients and calculate the required IDMS adjustments to keep an overall synchronization status. Therefore, it requires additional complexity in the Sync Clients. This can result in an increase of the development costs which can be seen as an important drawback of DCS.

An important limiting factor of DCS and M/S Scheme is the lack of support of multicast feedback capabilities (i.e., the ability of exchanging useful information for IDMS in a point-to-multipoint way) among the distributed Sync Clients in some media streaming technologies, such as those in which Single Source Multicast (SSM) with Unicast Feedback [Ott10a] is employed. In such cases, only the Distribution Source (entity defined in RFC 5760 [Ott10a]) can transmit data in a multicast way. So, it could prevent the deployment of an IDMS solution based on DCS or M/S Schemes in some actual large-scale environments, such as IPTV. An option here could be to send the IDMS reports to the Distribution Source in a unicast way, and then the Distribution Source forwards the incoming IDMS reports in a multicast way to all the other entities involved in the IDMS control process. However, this operation mode for IDMS adds extra traffic overhead and increases the latency. In other controlled scenarios, where small groups of users are consuming media content (e.g., watching TV) in a synchronized manner, independently of other users or groups of users, the adoption of DCS or M/S Scheme

[18] However, the traffic overhead added by the IDMS control messages should not be very high compared to the bandwidth consumption by the media stream to be synchronized.

would be a feasible option. This is not an issue when using SMS, as the Sync Clients typically send IDMS reports to the Sync Manager in a unicast way.

Finally, SMS is the best scheme in terms of consistency, coherence and security. Besides, this scheme can provide satisfactory responsiveness in terms of flexibility, traffic overhead, causality and fairness. Contrariwise, the main weaknesses of using SMS for IDMS are scalability and interactivity. Regarding scalability, there are no significant differences between SMS and DCS. Moreover, the performance in terms of scalability in SMS can be improved by using two control mechanisms: i) dividing the session members into logical groups, which facilitates the IDMS control to the Sync Manager; and ii) dynamically adjusting the transmission interval for the IDMS reports according to the number of active Sync Clients and to the available bandwidth (as done, for example, when using RTP/RTCP [Sch03] for streaming media). The second weakness (interactivity) is not a crucial drawback in those scenarios that do not require stringent synchronization levels (e.g., in shared video watching). For instance, previous works (e.g., [Bor08]) have shown the feasibility of SMS to keep the asynchrony within allowable limits in real scenarios (even more stringent synchronization levels that the ones required for Social TV were accomplished).

Besides, in some media streaming technologies, such as the ones using RTP/RTCP, distributed receivers regularly send feedback messages including QoS metrics (e.g., delay, jitter, packet loss information, etc.) to the Media Server, who can react accordingly (e.g., by adjusting its transmission timing or the media encoding mechanisms). If those feedback messages are extended to include useful information for IDMS, it would facilitate the deployment of an IDMS solution (as will be discussed in Chapter 8). This makes SMS the most practical alternative for most IDMS use cases, especially if the Sync Manager functionality is integrated within the Media Server resources.

Therefore, taking into account all the above features, it can be concluded that SMS is, in general, the best-ranked scheme for IDMS. In particular, SMS is preferable in those scenarios in which coherence is essential, the network delays are not excessively large, and the number of Sync Clients is not too high, such as networked loudspeakers, phased array transducers and sound reinforcement systems (in which a central entity responsible for mixing, filtering and prioritizing functions must be included). SMS is also adequate for on-line election events (in which all the votes must be registered in a central control entity), as well as for distributed shared video watching scenarios and video walls (in which feedback control reports are usually sent from the receivers to the Media Server for QoS monitoring purposes).

5.4 Summary

This Chapter has provided a thorough discussion about the suitability and applicability of both architectural approaches and control schemes for IDMS.

First, the advantages and disadvantages of network-based IDMS approaches compared to terminal-based IDMS approaches have been discussed. In particular, the main advantages of using network-based IDMS approaches are: higher scalability, lower zapping delays and lower complexity of the UEs. However, the deployment of network-based IDMS approaches also implies various limitations. Concretely, they require (a major) control of the network provider, they offer lower synchronization accuracy (since any delay variability between the edge nodes and UEs cannot be compensated) and they do not provide an efficient support when multiple logical groups of Sync Clients need to be simultaneously synchronized. Moreover, network-based approaches are better suited for IDMS-enabled sessions involving a large number of participants (e.g., IPTV and MOGs). However, in many IDMS use cases, the number of participants will not be very high, and they can further be divided in different groups. In such cases, the use of terminal-based IDMS approaches becomes more convenient.

Second, an exhaustive qualitative comparison among the IDMS control schemes has been provided. This study is relevant, because the adoption of a specific control scheme is a key decision when designing and developing an IDMS solution. Several criteria to determine the most appropriate control scheme in specific situations have been provided. Although it has been concluded that SMS is, in general, the best-ranked scheme for IDMS, the appropriateness of DCS and M/S schemes for specific use cases has also been identified. On the one hand, M/S Scheme outperforms SMS in terms of scalability, traffic overhead, interactivity and causality. On the other hand, DCS can provide better performance than SMS in terms of robustness, scalability, interactivity, flexibility and fairness. Therefore, the selection of a specific control scheme will strongly depend on the context and space in which an IDMS solution is going to be deployed.

Although no definitive rules have been provided, the thorough discussion about the appropriateness of each control scheme can be used as indicative guidelines by researchers interested in developing IDMS solutions. Indeed, the findings of this qualitative study have had an impact on the design of the IDMS solution presented in this PhD thesis. The initial premise was to uniquely base its design on the use of SMS. However, due to the identified convenience of using M/S Scheme and DCS in specific situations, their adoption in the designed IDMS solution (presented in Chapter 8) was also decided. This will allow to efficiently deploy our IDMS solution in a wide range of scenarios, according to the targeted use cases (e.g., Social TV, e-learning, audio beamforming…), the specific requirements (e.g., interactivity, scalability, accuracy, coherence…), and the characteristics and available resources of the networked environment (e.g., multicast support, delays, bandwidth…).

Chapter 6

KEY REQUIREMENTS FOR IDMS

6.1 Introduction

The previous Chapters have comprised the problem analysis phase of this PhD thesis (see Figure 6.1). It started with a study of the shared media consumption paradigm, by analyzing the relevant IDMS use cases and associated challenges (in Chapter 2). After that, a thorough review of the start-of-the-art allowed to identify the necessary components, and their alternatives, to achieve IDMS (in Chapter 3), as well as the strengths and weaknesses of existing IDMS solutions (in Chapter 4). As a result of such research stages, several (technical) requirements for IDMS have been derived, which are introduced in this Chapter.

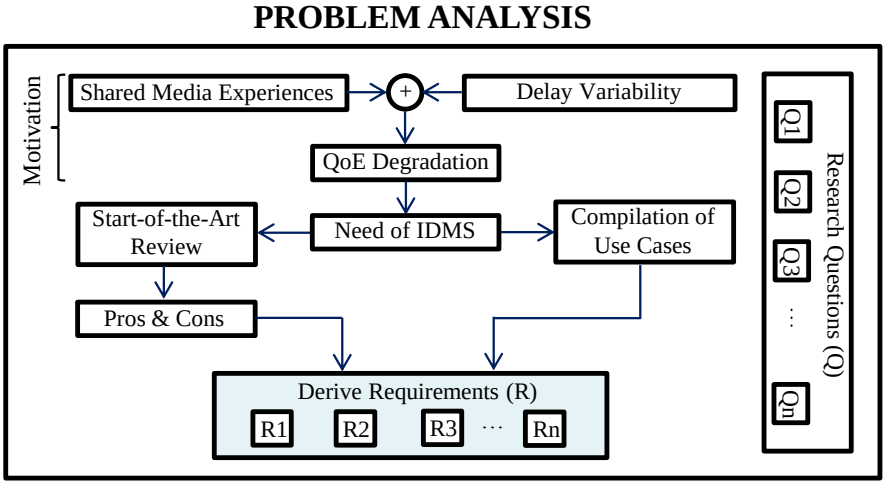


Figure 6.1. Problem Analysis Phase of this PhD thesis.

6.2 Key requirements for IDMS

This Section lists the derived requirements for IDMS. It is important to emphasize that all them must not be mandatorily supported by all IDMS solutions, but rather their compliance will be useful or beneficial for IDMS.

R1. Provisioning of metadata (e.g., timestamps, sequence numbers, source and media type identifiers…) in the media delivery units (at the server side) to enable the reconstruction of the original media timing (at the client side).

R2. Availability of an adaptive and scalable feedback control channel. This would enable a continuous monitoring and control of the IDMS process. This feedback channel, or an additional one, would also be very useful to negotiate and inform about key aspects for IDMS (e.g., group membership, clock sources, bootstrapping information…).

On the contrary, some IDMS solutions mainly rely on synchronizing specific control events or playout position updates (e.g., play, pause, seeking or stop commands), but do not provide continuous monitoring and control processes to achieve synchronization. Accordingly, such solutions provide a coarse synchronization process, with lower accuracy, because of the continuous and unpredictable end-to-end delay variability during a media session's lifetime.

R3. Compensation of the end-to-end delay (i.e., from capture/retrieval at the server side to presentation/playout at the client side) variability between the involved sync entities.

The compensation of the network delay variability, by enforcing the largest network delay to all the Sync Clients (as done, for example, in [Dio99], [Mau04] and [Vai11b]) is not sufficient, because media content is also significantly delayed in different manners at the client side, mainly due to different buffering settings, different hardware and software resources, variable CPU load, etc.

R4. Support of high synchronization accuracy. Some IDMS solutions are targeted at offering loose synchronization levels, but are unable to meet the stringent synchronization levels required in various IDMS use cases, as discussed in Section 2.4.

R5. Reliance on wall-clock timing mechanisms, as well as on control mechanisms for negotiating the use of common or related wall-clock sources between the involved sync entities. This would allow a coherent framework for stamping, interpreting an aligning timelines through the end-to-end media delivery chain.

R6. Combined usage of and coordination between various synchronization specification methods to leverage their joint potential for IDMS: axis-based (to provide continuous timelines as the main reference for synchronization),

control-based (to complement the axis-based approach with extra periodic synchronization metadata for accurately re-aligning timelines), and event-based (to be able of triggering dynamic synchronization adjustments as an early response to specific events).

R7. Adaptability in best-effort networked environments. The IDMS solution must perform well in best-effort environments, be adaptive to variable network conditions, and should not (exclusively) rely on reservation-based mechanisms.

R8. Support of rate adaption techniques (at the server-side, at the client-side, or both) to adjust the IDMS timing, allowing to dynamically maintain and restore the original synchronicity.

R9. Support for multiple media types (especially, audio and video) and encoding mechanisms. Additionally, the IDMS solution must be extensible to be able to accommodate emerging media types and encoding needs.

R10. Similarly, the IDMS solution must perform efficiently with both Constant Bit Rate (CBR) and Variable Bit Rate (VBR) streams. It must not be limited only to CBR streams, as some of the existing IDMS solutions surveyed in Chapter 4.

R11. Support of the IDMS protocols by both network and end-system entities. This would allow a future deployment of highly scalable IDMS solutions (e.g., for IPTV services [Sto10], or MOG) based on in-network processing.

R12. Support for both CoD and live media services.

R13. Valid for as many IDMS use cases as possible, unlike other ad-hoc or application-specific IDMS solutions, which are only applicable to single (or a small set of) media applications.

R14. Valid for both unicast and multicast transmission modes. In some scenarios, multicast is either not available or not optimal. However, multicast delivery may be preferable in large scale scenarios, with dynamic group membership.

R15. Flexibility to be efficiently and reliably deployed in various networked scenarios, with different requirements and available resources (e.g., bandwidth, latency, multicast support…). Likewise, single points of failure should be avoided or overcome, when possible.

R16. The IDMS solution must rely as much as possible on existing, and standard (if possible), technologies.

The existing IDMS solutions define new proprietary protocols, with their own defined control messages, that may increase the network load. Such solutions proved to perform satisfactorily, but mainly in (vendor) walled-garden or controlled environments. However, the reliance on proprietary

solution makes compatibility between (third-party) implementations difficult, especially when deployed in large-scale (uncontrolled) scenarios, involving third-party infrastructure and communication devices.

A standardized, or at least standard compliant, IDMS solution, based on existing technologies, will assure inter-operability and help the uptake of implementations, ensuring a more widespread support and adoption of IDMS in practice. Standardization can also keep costs down, allowing vendors to have a bigger potential market for their products. This is especially relevant in IPTV systems.

6.3 Summary

This Chapter has presented several key requirements for IDMS, which have been derived as a result of the problem analysis phase of this PhD thesis. The provided list of requirements will be basis on which the IDMS solution will be designed. Moreover, these findings can also be used by other researchers and developers to help identifying the features that should be provided when deploying IDMS-related technology.

Chapter 7

RTP/RTCP PROTOCOLS

7.1 Introduction

In this Chapter, an overview of the capabilities of RTP/RTCP protocols to enable multimedia synchronization is provided. First, the relevance of such protocols in the current media consumption paradigm is discussed in Section 7.2. Next, Section 7.3 briefly explains the transport-level configurations supported in multimedia sessions using RTP/RTCP protocols. In Section 7.4, the RTP/RTCP features, specified in RFC 3550 [Sch03], that are useful to enable intra-media and inter-media synchronization are described. Finally, in Section 7.5, a summary of the RTCP feedback reporting rules specified in different IETF standards is provided.

7.2 Multimedia Delivery Protocols: Relevance of RTP/RTCP

Undoubtedly, the delivery of media content over Internet has gained an increased relevancy during the last years, and this is expected to grow further in the near future ([Beg11a], [Li13], [Swa13]). CoD, IPTV, audio/video conferencing and Social TV are just a few examples of this boom of media streaming services.

Realizing successful media delivery services requires the optimization of, and coordination between, many technological aspects ([Li13], [Swa13]), such as encoding techniques, network infrastructures, protocols, signaling mechanisms, control techniques, and so on. In particular, substantial research efforts have been devoted on devising proper broadband delivery protocols to overcome various challenges in terms of latency, bandwidth, deployment costs, adaptability, scalability, etc. Currently, media delivery can be accomplished via a blend of choices between downloading and streaming solutions, being the latter more widely adopted ([Mai09], [Beg11a], [Li13], [Swa13]). Two main forms of streaming services can be distinguished: managed and unmanaged. Managed services, such as cable TV and IPTV, are quoted services that mainly operate within walled-garden

(geographically restricted and privately owned) environments, with QoS guarantees. This kind of services mainly rely on push-based RTP over UDP streaming, either in a unicast or multicast way, and typically provide service-compliant media delivery, including protection, authentication and re-transmission mechanisms [Beg11a]. The main benefits of RTP streaming are low latency, interactivity, and bandwidth efficiency. Moreover, the feedback mechanisms provided by RTCP provide many relevant features, such as QoS monitoring, participants' identification and synchronization. For instance, the RTCP reporting features can be used by IPTV service providers for troubleshooting or fault tolerance management [Beg10]. RTP/RTCP protocols require the availability of dedicated stateful servers and the installation of either (commercial or open-source) media players or STBs at the client side. Currently, RTP/RTCP protocols are used in a plethora of CoD services, even though such protocols are especially suited for real-time interactive services, such as IPTV and audio/video conferencing. Contrarily, unmanaged or OTT Internet services, such as WebTV, are free services that can operate worldwide (along public non-managed IP-based scenarios). This kind of services mainly employ unicast pull-based Hypertext Transfer Protocol (HTTP) over TCP streaming, in a best-effort basis. The main advantages of HTTP streaming are scalability, reliability, reachability and deployment cost efficiency. Unlike RTP streaming, HTTP streaming does not require a one-to-one session between the individual clients and the server, so it allows: i) scaling to a large number of users; ii) not keeping dynamic UDP ports open, thus overcoming firewalls and NAT (Network Address Translation) traversal issues; and iii) leveraging the conventional web infrastructure, such as stateless HTTP servers, as well as CDN and Internet Service Provider (ISP) caches. Another relevant issue is that HTTP streaming only requires a standard web browser to consume media content, so there is no need of installing (third-party) media players or STBs. In this context, different vendors and standardization bodies have specified their own HTTP streaming solution, such as: HTTP Live Streaming (HLS) by Apple, HTTP Dynamic Streaming (HDS) by Adobe, Microsoft Smooth Streaming Protocol (MS-SSTR) by Microsoft, and MPEG Dynamic Adaptive Streaming over HTTP (DASH) by ISO/IEC and MPEG. The latest advances in HTTP streaming have led to a clear trend towards the deployment of this web-based technology for unidirectional (on-demand) media content delivery. As proofs of evidence, MPEG DASH has been recently adopted by the Hybrid Broadcast Broadband TV (HbbTV) [19] standard, and by various popular media services, such as Netflix, Hulu, YouTube and Akamai.

Nevertheless, advances have not only be devised for HTTP streaming, but also for RTP streaming. Since their inception, the RTP/RTCP functionalities have been continuously improved and extended upon to accommodate emerging

[19] Hybrid Broadcast Broadband TV (HbbTV), http://www.hbbtv.org/.

requirements[20]. Nowadays, RTP/RTCP provide several key features, such as scalability, security, multiplexing, compression, error control and congestion control. In particular, two key improvements aim to promote the deployment and spread the ubiquity of RTP streaming. On the one hand, the multiplexing features (briefly explained in next Section) allow easing NAT traversal and simplifying firewalls administration. On the other hand, the development of WebRTC (Web Real Time Communications) [Lor12] has also allowed a native integration of RTP/RTCP protocols into traditional web browsers, thus eliminating the need of (third-party) media servers/players. WebRTC is a revolutionary and promising web-based communication model that supports various forms of (multi-party) conferencing services [Hol14].

Given the evolution and the different strengths and weaknesses of both streaming-like solutions, the idea that a single protocol can efficiently meet all the requirements for successful delivery systems (i.e., a "one-size-fits-all" approach) was rapidly dismissed, as each of them are best suited for specific use cases ([Beg11a], [Swa13]). The relevance of RTP/RTCP in current and future media deployments is beyond doubt, especially in those scenarios in which tight real-time requirements must be met, and interactivity between users and the media content, as well as between users, is pursued.

7.3 RTP/RTCP Delivery and Multiplexing Features

Typically, each RTP stream conveys a specific media type (e.g., audio, video…) and, in conjunction with its associated RTCP packets, is carried in a separate RTP session, which is defined as an association among a set of participants communicating via RTP. Therefore, in a multimedia session, each participant may be involved in multiple RTP sessions at the same time (i.e., one RTP session per each involved media type). This enables the receivers to process only the particular RTP stream they are either interested in (or they are able to). The participants in an RTP session may share common destination transport addresses (i.e., same IP addresses and UDP ports), which can be negotiated via other protocols such as Real Time Streaming Protocol (RTSP) [Sch98], using Session Description Protocol (SDP) [Han06] in the RTSP *Setup* method, and Session Initiation Protocol (SIP)

[20] The core functionalities of RTP/RTCP are specified in RFC 3550 (which obsoletes RFC 1889), but many other IETF standards have specified modifications, improvements and/or extensions to such protocols. Moreover, although the IETF is the organization responsible for standardization of RTP/RTCP, such protocols have also been adopted by other standardization organizations, such as the International Telecommunication Union (ITU), the European Telecommunications Standards Institute (ETSI), the World Wide Web Consortium (W3C), the 3rd Generation Partnership Project (3GPP) and the Open IPTV Forum (OIPF).

[Ros02]. According to the RFC 3550, an even port number should be used for RTP, while the next higher odd port number should be assigned to RTCP.

However, other configurations are possible. For example, the multiplexing of several media types onto a single port is also supported in RFC 3550 [Sch03]. Likewise, RFC 5761 [Per10a] discusses issues that arise when multiplexing RTP and RTCP packets on a single transport address to ease NAT traversal and simplify firewall administration. RFC 5761 also describes use cases when such multiplexing is or is not appropriate. Moreover, the upcoming standard in [Len14] clarifies and expands RFC 3550 with the intention of providing better support for use cases in which the transmission of different RTP streams within single RTP sessions is beneficial. Examples are scenarios involving arrays of capturing devices (e.g., video cameras and/or microphones), involving multi-stream mixers, or when scalable video coding mechanisms are used.

7.4 RTP/RTCP for Intra-Media and Inter-Media Synchronization (RFC 3550)

This section provides an overview of the features of RTP (sub-section 7.4.1) and RTCP (sub-section 7.4.2) specified in RFC 3550 that are useful to enable, or have an impact on, intra-media and inter-media synchronization.

7.4.1 RTP (Real-Time Transport Protocol)

RTP provides a framework for delivery of time-sensitive media content between distributed end-systems. It defines a malleable payload-dependent framing level for conveying media, by packetizing the encoded application-layer MUs into RTP packets. If the MUs are large (e.g., video frames), they may be fragmented into several RTP packets, whereas if the MUs are rather small (e.g., audio samples), several of them may be bundled into a single RTP packet.

RTP typically runs on top of UDP, either in a unicast or multicast way, even though there is no restriction to use RTP on top of TCP, which indeed can be useful to avoid firewall and NAT issues when streaming over Internet. RTP provides an enhancement to the transport level, by adding useful features, such as media type and source identification, synchronization and loss detection. However, it is important to point out that RTP does not guarantee (timely) delivery of packets, but the responsibility for dealing with late and/or lost packets is left to the application-layer.

Regarding multimedia synchronization, the RTP packet's header includes the following useful metadata: i) synchronization source (SSRC) identifier; ii) sequence number; iii) generation timestamp; iv) Payload Type (PT) identifier; and v) Marker (M) bit.

The *SSRC identifier* field (32-bit integer) allows for uniquely identifying RTP sources within a media session. It is randomly generated by RTP (media) sources when joining the session. If a participant generates multiple streams - e.g., from separate video cameras -, each RTP stream must have a different SSRC identifier. This way, the incoming RTP packets with the same SSRC identifier must be grouped for media playout at the receiver side. The use of SSRC identifiers is a better mechanism to identify RTP streams than the use of underlying transport parameters (i.e., IP addresses and ports numbers), which can vary throughout the duration of the session and do not necessarily identify the original source of the RTP packets (e.g., when RTP mixers or translators are involved). Even though it is very unlikely that two RTP sources generate the same SSRC identifier, every RTP implementation should implement a mechanism to cope with this chance.

The *RTP sequence number* field (16-bit unsigned integer) identifies each individual RTP packet within a specific RTP stream. It is given by a counter, initialized from a random number (this helps avoiding known-value decryption attacks in case that RTP end-systems encrypt the streams), which increases by one in each generated RTP packet and wraps around to zero when the maximum value is reached. The use of sequence numbers provides two key benefits: i) it allows reconstructing the original order in which RTP packets were sent; and ii) it allows detecting packet loss.

The *RTP timestamp* field (32-bit unsigned integer) denotes the sampling instant of the first octet of data in each RTP packet, and it is very useful for scheduling the media playout at the receiver side. The initial value of the RTP timestamp is also randomly generated and it wraps around to zero upon reaching the maximum value. If MUs are fragmented into multiple RTP packets, all of these packets will have the same timestamp (as all of them will contain data captured/sampled at the same instant), but will differ in their sequence number. The RTP timestamp is derived from a local clock that must increase in a linear and monotonic fashion (except for wrap-around, of course), producing a single and independent timeline for each RTP stream. This local clock is commonly provided by the corresponding hardware or software components for media capture/retrieval for each media stream (e.g., the audio/video input/capturing cards).

In some cases, the order in which media is sampled/captured is different to the order in which media is transmitted over the network. A typical example is when using MPEG video encoding, involving key frames (I-frames) and delta-encoded frames predicted from them forward (P-frames) and backward (B-frames). In such cases, video frames are assigned a timestamp once being captured, but their transmission can be delayed, because other future video frames may depend on such frames. This will result in an RTP stream with non-monotonically increasing timestamps, even though the sequence number order is always kept. Receivers will be responsible for reconstructing the timestamp order to properly play out the media.

The specific nominal rate of the (local) clock used to generate the RTP timestamps is payload-dependent (it must be equal or higher than the media sampling rate). That is why well-defined mechanisms are needed to inform about this value, as well as about additional parameters of the media to be delivered. In conjunction, this set of parameters will determine how to encapsulate, transport and interpret (at the receiver side) each RTP stream when specific media types or payload formats are employed. Examples are: the type of media (audio, video …), encoding mechanisms, number of channels (for audio), sampling clock rate, packet size, frame size (fixed or variable), if additional packet headers are needed, etc. This set of parameters constitutes the media profile, which is typically described, announced and/or negotiated by using out-of-band mechanisms, such as via SDP. For example, the *RTP Audio Video Profile* (RTP/AVP) is specified in RFC 3551 [Sch03b]. The RTP/AVP provides an association between specific RTP payload formats (a set of encoding and encapsulation mechanisms) and a *Payload Type (PT) identifier*, which is an 8-bit integer field included in the header of each RTP packet. Different RTP payload formats are defined in RFC 3551, in which the processes of defining and registering future RTP payload formats (e.g., when new codecs are designed) are also described.

RTP receivers can determine specific characteristics of the incoming RTP streams, such as the encoding type or the clock rate of RTP timestamps, by looking at the value of the PT identifier of the RTP packets. In some cases, the mapping between the RTP payload format and the PT identifier is static; in other cases, the mapping is dynamically negotiated via out-of-band mechanisms (e.g., via SDP). For RTP payload formats with static PT assignments, the clock rate is implicit (i.e., it is specified as part of the PT assignment). However, the dynamic PT assignment process must explicitly specify the clock rate along with the payload type (since different options can be selected for that in specific encoding mechanisms).

As many RTP payload formats were being specified since the inception of RTP/RTCP protocols, the usage of a static mapping for each one of them would have had to involve large and complex mapping tables (besides making an 8-bit field to represent the PT identifier insufficient). In addition, most of the current RTP payload formats require some previous signaling and negotiation phases to reach an agreement on the settings of the supported parameters. Therefore, the usage of dynamic PT assignments rapidly became the preferred solution.

If the conditions of the network and/or end-systems vary during an on-going session, it is also possible to dynamically change the RTP payload format (e.g., by adjusting the media encoding settings), and then including thereafter the new associated PT identifiers in the transmitted RTP packets to notify about that.

Table 7.1 provides some of the most common examples of mapping between RTP payload formats and PT identifiers, in conjunction with the specific media

types, codecs, the supported clock rates, and the IETF standards in which they are specified.

Finally, the *Marker (M)* bit is used to inform about events of interest within media streams. Its precise meaning is defined by the RTP profile and media type in use, but it can also be relevant for multimedia synchronization. For example, in audio streaming, the M bit can be set to 1 to indicate the first RTP packet sent after a period of silence, and otherwise set to 0. This indication can be useful to trigger playout adjustments during silence periods, because a small variation in the length of a silence period is usually unnoticeable to listeners, whereas a playout adjustment when there is audio activity can be annoying to them. However, this would not be applicable when streaming music, since silence periods are almost as important as activity periods in such a case. Likewise, in video streaming, the M bit can be set to 1 to indicate the last RTP packet containing a video frame (i.e., the packet with the highest sequence number), and otherwise set to 0. This way, the M bit can serve as an indication to start decoding a video frame, provided that all the previous RTP packets have been already received.

Finally, it is important to mention that despite the local clock rate is either signalized out-of-band (e.g., via SDP) or implicitly given by the specific PT identifier, the RTP specification does not provide guarantees about the resolution, accuracy or stability of the media clocks. Therefore, any differences (i.e., skews) between the nominal clocks of the senders and receivers, or drifts of the individual clocks, may cause non-smooth playout and loss of synchronization. These differences must be compensated by using additional clock adjustment algorithms.

Table 7.1. Examples of RTP/AVP and their Mapping to PT Identifiers

Payload type (PT)	Name	Type	Clock rate (Hz)	IETF Standard
0	PCMU	audio	8000	RFC 3551
3	GSM	audio	8000	RFC 3551
8	PCMA	audio	8000	RFC 3551
26	JPEG	video	90000	RFC 2435
33	MP2T	audio/video	90000	RFC 2250
34	H263	video	90000	RFC 2190, RFC 3551
dynamic	H264	video	90000	RFC 6184
dynamic	iLBC	audio	8000	RFC 3952
dynamic	G719	audio	48000	RFC 5404
dynamic	AMR	audio	8000	RFC 4867
dynamic	vorbis	audio	any (must be a multiple of the sample rate)	RFC 5215
dynamic	speex	audio	8000, 16000, or 32000	RFC 5574
dynamic	MP4V-ES	video	90000, or others	RFC 6416

7.4.2 RTCP (RTP Control Protocol)

The RTP data transport protocol is augmented by RTCP protocol. RTCP provides an adaptive feedback channel between participants in a media session, which allows exchanging useful statistics about the RTP data delivery, monitoring the session membership, identifying participants, enabling synchronization, as well as conveying other relevant information regarding the media session.

Five types of RTCP packets were initially defined in RFC 3550: Receiver Report (RR), Sender Report (SR), Source Description (SDES), BYE, and Application-Defined (APP). Such RTCP packet types and their purpose are listed in Table 7.2. Their format and a complete explanation of their fields can be found in RFC 3550. In this Chapter, their relevant features for multimedia synchronization are described.

Table 7.2. RTCP Packets Types (RFC 3550)

Packet's Name	Payload type (PT)	Sent by	Metrics or Purpose
Sender Report (SR)	200	Senders	Number of RTP packets and bytes sent so far. Mapping between RTP and NTP timestamps.
Receiver Report (RR)	201	Receivers	Fraction and cumulative number of packets lost, jitter, and useful data for round trip delay calculation.
Source Description (SDES)	202	Senders and Receivers	List of items including information about participants.
Goodbye (BYE)	203	Senders and Receivers	Notification that a participant has left the session.
Application-Defined (APP)	204	It depends on the application	Used for experimental purposes and vendor-specific applications.

RTCP Sender Report (SR) packets are sent by active RTP senders. These packets convey statistics about the media being sent, such as the total number of RTP packets and octets transmitted since the beginning of the session. Most importantly, each SR also contains a correspondence between its originating 32-bit RTP timestamp (obtained from a local clock) and its originating 64-bit NTP-based timestamp (obtained from a global clock, e.g. provided by NTP [Mil10]). On the one hand, this mapping time information will allow checking, and correcting, any inconsistencies between the local clocks of the sender and receivers, thus enabling intra-media synchronization. This is because both sender and receivers can use this (more accurate) "common" global wall-clock as a reference base time for multimedia synchronization. On the other hand, it will allow aligning several related RTP streams in the time domain at the receiver side, thus enabling inter-media synchronization. This is because the independent local timelines of each RTP stream can be mapped to this "common" global wall-clock.

Although the wall-clock timestamp in RTCP SR packets has an NTP-based format, the sender clock does not have to necessarily be synchronized with an external NTP source or have any particular accuracy, resolution, or stability. If RTP senders and receivers do not use (because they probably do not have access to) the same wall-clock server, any inconsistencies between the absolute timelines provided by these alternative reference clocks (e.g., system-specific clocks) can cause intra-media synchronization problems. Still, this is more accurate than only relying on local RTP timestamps. Synchronization between sender and receiver clocks is not indispensable for enabling intra-media synchronization (as the local RTP timestamps are sufficient for that), but can help to improve the synchronization accuracy in case the local clocks of sender and receivers drift over time.

Regarding inter-media synchronization, another key issue is the ability of associating the SSRC identifiers of the RTP streams to be synchronized. To achieve this, the different (intra-session) SSRC identifiers need to be linked to a common (inter-session) canonical and persistent identifier for each participant (it is important to remember that participants can send several RTP streams, each one with a different SSRC identifier). The *RTCP Source Description* (SDES) packets are used to convey such information, as well as additional details. Several types of SDES items are defined in RFC 3550, which allows the definition of additional ones in future standards. Examples of existing items are: CNAME (participant's canonical name), NAME (participant's name), EMAIL (participant's e-mail), PHONE (participant's phone number), LOC (participant's location), TOOL (name of the media application or tool being used), NOTE (information states or notes), and PRIV (used to convey experimental, private or application-specific extensions). Among them, the CNAME is the only mandatory item[21], which provides the necessary binding across the multiple RTP streams to be synchronized. Through this stable and persistent CNAME identifier (the SSRC identifier is randomly generated and will change if an application restarts or if an SSRC collision occurs), receivers can identify the different RTP streams that need to be synchronized. The CNAME is allocated algorithmically from the users' name and the IP address of their host, having a format *user@hostIP* (e.g., *mario_montagud@158.42.1.2*). This implies that, when using private IP addresses, NAT gateways will have to translate the CNAME identifier in a consistent way for all the involved RTP streams.

By using the information provided by each RTCP SR (correlation between RTP and NTP-based timestamps) and SDES packets (association between SSRC identifiers and CNAME items), receivers will be able to synchronize related RTP streams.

[21] The CNAME item can also be signaled out-of-band, e.g. via SDP, as specified in RFC 5576 [Len09].

The CNAME identifier is also relevant when streaming layered media (e.g., H.264 Scalable Video Coding - SVC - and MPEG surround multi-channel audio). In such cases, the involved layers are typically sent in different RTP streams (with different SSRC identifiers), and the (common) CNAME is the provided mechanism to enable their synchronization.

In order to help understanding the concepts introduced so far, Figure 7.1 illustrates an example of a transmission of two related RTP streams. The first one is an RTP stream containing audio captured by a microphone (left side), while the second one is an RTP stream containing video captured by a camera (right side). The RTP packets of the audio and video streams are represented in red and blue square boxes, respectively. It can be observed that the sequence numbers of both RTP streams are continuously increasing. This is also true for timestamps, with the exception of when several RTP packets contain the same MU, as it happens for the first two RTP packets of the video stream. It can also be observed that the timestamps of each RTP stream are obtained through a local clock, which can be provided by the audio/video input cards and runs at a specific sampling/capturing rate. In order to associate both independent RTP streams, RTCP is used. On the one hand, RTCP SRs packets (represented in circles in the figure) contain the relationship between the independent local clocks of each stream (i.e., RTP timestamps), represented at both sides of the figure, and a common (global) wall-clock (i.e., NTP-based timestamps), represented in the middle of the figure. This allows aligning both streams in the time domain according to this common timeline. On the other hand, RTCP SDES packets contain the binding between the individual SSRC identifiers of each RTP stream ($SSRC_A$ and $SSRC_V$ in the figure) and a common CNAME item ($CNAME_C$ in the figure).

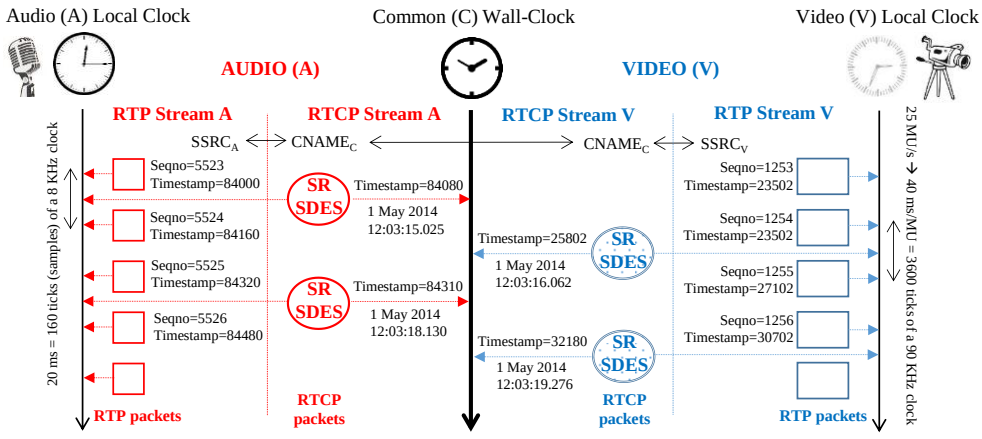


Figure 7.1. RTP/RTCP features for intra-stream and inter-stream synchronization.

RTCP Receiver Report (RR) packets are sent by RTP receivers to inform about QoS metrics for a specific RTP stream. An RR packet contains the SSRC of the participant sending the report, the SSRC of the source of the RTP stream this report refers to, and a set of additional fields. First, RRs contain information about lost packets. On the one hand, the *fraction lost* field (8-bit) indicates the number of lost RTP packets divided by the number of expected packets (according to the highest RTP sequence number received so far) since the transmission of the previous RTCP RR. On the other hand, each RR also reports on the *cumulative number of packets lost* (24-bit) since the beginning of the session. Second, the *extended highest RTP sequence number* (32-bit) that has been received is also included. Note that the sequence numbers in RTP packets have a length of 16 bits. This value is included in the lower part of the *extended highest RTP sequence number* field, whilst the most significant 16 bits include the corresponding count of sequence number cycles. This helps avoiding wrap around to zero issues and possible resets of the sequence numbering process. Third, the *inter-arrival jitter* field (32-bit unsigned integer) allows reporting on an estimation of the statistical variance in network transit times for the RTP packets. The inter-arrival jitter is calculated/updated upon receiving each RTP packet. In particular, if t_n represents the RTP Timestamp of *n-th* RTP packet, and r_n represents its arrival time, in RTP timestamp units, the time difference (or delay variation) for two consecutive *(n-1)-st* and *n-th* packets, $v_{n-1,n}$, is computed as:

$$v_{n-1,n} = (r_n - r_{n-1}) - (t_n - t_{n-1}) = (r_n - t_n) - (r_{n-1} - t_{n-1}) \quad \textbf{Eq. 7.1}$$

Then, this delay variation (or jitter) is smoothed according to Eq. 7.2, which gives only a small weight to the most recent observation to deal with temporary fluctuations:

$$j_n = 15/16 \cdot j_{n-1} + 1/16 \cdot v_{n-1,n} \quad \textbf{Eq. 7.2}$$

Finally, two additional fields are included to allow measuring the round-trip delay between the RTP sender and receivers. First, the *Last Sender Report timestamp* (LSR) field includes the middle 32-bits of the NTP timestamp included in the last received RTCP SR. Second, the *Delay since Last Sender Report* (DLSR) field (32-bits) includes the delay between receiving the last SR and sending the current RR, expressed in units of 1/65536 s. If no SR has been received yet, both fields are set to zero. With this information, the RTP source can calculate the round-trip delay to each receiver upon receiving a new RR from that receiver, as shown in Figure 7.2. Regarding the measurement of this last QoS metric, it is important to reflect two issues: i) symmetric network delays are assumed, which is not always true; and ii) delays are somehow dependent on the size of the packets being transmitted and on the data transmission rate. Therefore, delays for RTCP packets

can only give a rough estimation of the delays for RTP packets, which are the ones that actually need to be controlled (or even bounded).

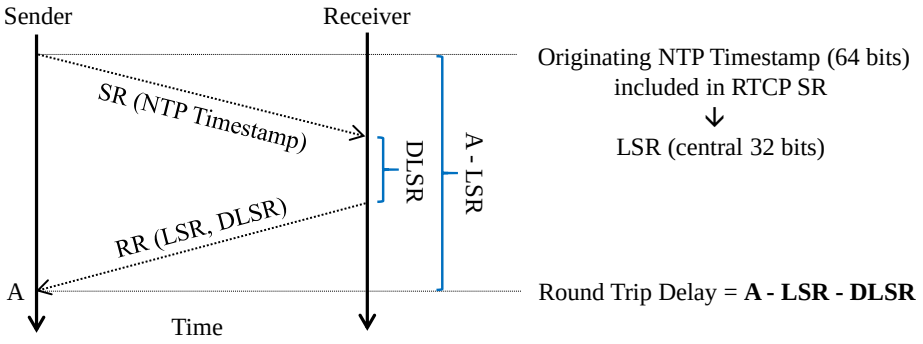


Figure 7.2. Round Trip Delay Measurement by using RTCP SR and RR packets.

Each one of the fields included in RR packets has an impact on multimedia synchronization, since an increase of delays, jitter or packet loss, may reflect (local or global) congestion situations and the consequent need for adjusting the transmission and/or playout processes.

RTCP BYE packets are generated when participants leave the session. They may also include a textual description indicating the reason for leaving. When BYE packets are received, the participants' database is updated.

Finally, *RTCP APP packets* allow for application-defined extensions. These packets are aimed at experimental purposes, as a first glance for testing with new features, but are not suited for defining standard compliant extensions to RTCP. New RTCP reports or packets must be used for that, as specified in RFC 5968 [Ott10b]. Any RTP implementation should be prepared to ignore unrecognized APPs, as some applications may generate their own defined APP packets.

The RTCP packets are generally sent in compound RTCP packets (i.e., several individual RTCP packets are grouped together for their transmission), even though RFC 5506 [Joh09] also specifies certain situations in which it is possible, and suitable, to send RTCP packets individually.

Likewise, RTCP packets are generally much smaller than RTP ones (as they are control packets and do not contain media data) and are sent much less frequently. Whilst RTP packets are typically sent every few milliseconds (depending on the data transmission rate), RTCP packets will be exchanged on a scale of a few seconds, or even of minutes in large scale sessions. In the next Section, the RTCP reporting rules are described.

The feedback provided by RTCP packets may be used to trigger adaptation mechanisms during the media session's lifetime. On the one hand, the information of RTCP packets from senders can cause receivers to adjust their media playout rate or buffer settings (e.g., based on synchronization information). On the other hand, the information of RTCP packets from receivers is useful to determine if senders have to adjust their transmission rate or specific encoding settings (e.g., based on an increase of the round trip delays, jitter, or packet loss).

7.5 RTCP Reporting Rules

In this section, an overview of the RTCP reporting rules specified in different IETF standards is provided. The content of this section is important to help understanding two key features of the IDMS solution under design in this PhD thesis (presented in Chapter 8). The first one is the adaptability of the RTCP feedback rate according to the population of the session and the available bandwidth. The second one is the rationale and the basis on which the EED RTCP reporting rules for IDMS (presented in Chapter 9) have been designed, as well as the provided benefits by such mechanisms.

7.5.1 Regular RTCP Feedback (RFC 3550)

During the media session's lifetime, the participants of an RTP Session (i.e., senders and receivers) regularly exchange RTCP reports (typically conveyed into compound RTCP packets) to mainly inform about QoS statistics [Sch03]. On the one hand, a low frequency of RTCP feedback reporting can lead to faulty behavior due to outdated statistics. On the other hand, excessive reports can be redundant and cause unnecessary control traffic, probably leading to potential congestion situations. Also, if the RTCP packets were exchanged at a constant rate, the control traffic would grow linearly with the number of participants. Accordingly, a trade-off between up-to-date information and the amount of control traffic must be met. This would allow an application to (automatically) scale over session sizes ranging from few participants to tens of thousands.

The total amount of control traffic added by RTCP should be limited to a small (so that the primary function of media data transport is not impaired) and known (so that each participant can independently calculate its share) percentage of the allocated RTP session bandwidth. A fraction of 5 % is recommended in RFC 3550. In such process, media senders are given special consideration to allow a more frequent report exchange of their RTCP statistics, some of which are indeed very relevant for multimedia synchronization. In particular, if the proportion of senders constitute less than one quarter of the session membership (i.e., $n_{senders} \leq \frac{1}{4} \cdot n_{participants}$, where $n_{participants} = n_{senders} + n_{receivers}$), this percentage is further divided into two parts, where 25 % must be dedicated to active senders and the remaining can be consumed

by receivers. Otherwise, the RTCP bandwidth is equally shared between senders and receivers.

Based on the above aspects, the RTCP report interval, $T^{3550}_{RTCP\_d}$, is dynamically and deterministically computed in each RTP entity, every time an RTCP packet is sent, according to the estimation of the available session bandwidth ($BW_{session}$), the average size of all sent and received RTCP packets ($RTCP_{size}$), the number of participants in the session, their role (senders or receivers), as well as the unicast or multicast nature of the session. Such process is shown in Eq. 7.3:

$$\text{if } (n_{senders} \leq \frac{1}{4} n_{participants}) \begin{cases} \text{senders:} & T^{3550}_{RTCP\_d} = \dfrac{n_{senders} \cdot avg(RTCP_{size})}{0.25 \cdot BW_{RTCP}} \\ \text{receivers:} & T^{3550}_{RTCP\_d} = \dfrac{n_{receiverss} \cdot avg(RTCP_{size})}{0.75 \cdot BW_{RTCP}} \end{cases}$$

$$\text{else,} \quad T^{3550}_{RTCP\_d} = \frac{n_{participants} \cdot avg(RTCP_{size})}{BW_{RTCP}} \qquad \textbf{Eq. 7.3}$$

$$\text{, being } \; BW_{RTCP} = 0.05 \cdot BW_{Session}$$

However, $T^{3550}_{RTCP\_d}$ should have a lower bound to avoid having bursts of RTCP packets. The recommended value in RFC 3550 for the minimum interval, $T^{3550}_{RTCP\_d\_min}$, is 5 s. Besides, a delay should be imposed to each participant before sending the first RTCP packet upon joining the session. This allows a quicker convergence of the RTCP report interval to the correct value. This initial delay may be set to half the minimum RTCP report interval (i.e., 2.5 s) in multicast sessions, whilst it may be set to zero in unicast sessions. In some cases (e.g., if the data rate is high and the application demands more frequent RTCP reports), an implementation may scale $T^{3550}_{RTCP\_d\_min}$ to a smaller value given by 360 divided by $BW_{session}$ (in kbps). This yields an interval smaller than 5 s when $BW_{session}$ becomes greater than 72 *kbps*. In multicast sessions, only active senders may use that reduced minimum interval, whilst in unicast sessions it also may be used by receivers. Accordingly, the minimum value between $T^{3550}_{RTCP\_d}$ and the selected option for $T^{3550}_{RTCP\_d\_min}$ will be used for the RTCP report interval:

$$T^{3550}_{RTCP\_d\_\min} = \min\left(T^{3550}_{RTCP\_d\_\min}, T^{3550}_{RTCP\_d}\right) \qquad \textbf{Eq. 7.4}$$

In the above cases, however, the minimum interval of 5 s must be still taken into account during the membership accounting procedure to not prematurely time out participants (who can indeed be using it) because of inactivity.

After that, the interval between RTCP packets is varied randomly over the range [0.5, 1.5] times that minimum RTCP report interval ($T^{3550}_{RTCP\_d\_min}$) to prevent floods of RTCP reports (i.e., to avoid that all RTCP packets are sent and received almost at the same time, in every report interval):

$$T^{3550}_{RTCP\_random} = (0.5 + rand()) \cdot T^{3550}_{RTCP\_d\_\min} \in [0.5, 1.5] \cdot T^{3550}_{RTCP\_d\_\min}$$
$$\text{where: } rand() \in [0, 1] \qquad \textbf{Eq.7.5}$$

Additionally, *"timer reconsideration"* algorithms are introduced to allow for a more rapid adaptation of the RTCP report interval in large-scale sessions, where the membership can largely vary (e.g., many receivers join and leave the session quite frequently). To compensate for the fact that the *"timer reconsideration*" algorithms converge to a lower value than the intended average RTCP bandwidth, the (randomized) report interval is finally divided by e-3/2=1.21828:

$$T^{3550}_{RTCP\_recons} = T^{3550}_{RTCP\_random} / (e - 3/2) \qquad \textbf{Eq. 7.6}$$

7.5.2 Early RTCP Feedback (RFC 4585)

In RFC 4585 [Ott06], further RTCP reporting mechanisms are specified to enable receivers to provide, statistically, more immediate RTCP feedback to the senders. This Early RTCP Feedback profile, which is known as *RTP Audio-Visual Profile with Feedback* (RTP/AVPF), allows for short-term adaptation and efficient feedback-based repairing mechanisms to be implemented, while maintaining the RTCP bandwidth constraints and preserving scalability to large groups.

The RTCP report interval specified in RFC 3550 is denoted as Regular RTCP interval in RFC 4585. In addition, RFC 4585 enables to send RTCP reports earlier than the next scheduled Regular RTCP transmission time if a receiver detects the need to inform about media stream related events (e.g., picture or slice loss) close to their occurrence[22].

The reporting rules for Regular RTCP packets in RFC 4585 are similar than the ones in RFC 3550. However, $T^{3550}_{RTCP\_d\_min}$ is dropped in RFC 4585. Instead, an optional attribute, called *trr-int*, is specified as an offset parameter (in ms) to $T^{3550}_{RTCP\_d}$:

$$T^{4585}_{RTCP} = (0.5 + rand()) \cdot ((trr\text{-}int) + T^{3550}_{RTCP\_d} / (e - 3/2)) \qquad \textbf{Eq. 7.7}$$

[22] A suppression mechanism is adopted, in which receivers wait for a random dithering interval to avoid RTCP feedback implosion (i.e., lots of receivers reporting on the same event at the same time).

Note that providing *trr-int* as an independent variable is intended to restrain from sending too frequent *Regular RTCP packets* (i.e., saving RTCP bandwidth) while enabling higher flexibility to transmit *Early RTCP packets* (i.e., using the saved RTCP bandwidth) in response to dynamic events. This could not be achieved by reducing the overall RTCP bandwidth, because the frequency of Early RTCP packets would be affected as well. Values between 4 and 5 s for *trr-int* are recommended to assure inter-working with RTP entities only using Regular RTCP Feedback. However, as *trr-int* is an optional attribute, it may be set to zero (default value) if a specific application would benefit from a higher frequency of Regular RTCP packets. In such a case, the only difference between the RTCP timing rules from RFC 3550 and RFC 4585 for transmitting Regular RTCP packets resides in the minimum value for the report interval, which is dropped in RFC 4585.

In order to preserve the RTCP traffic bounds, only one Early RTCP packet can be transmitted between two consecutive Regular RTCP packets (i.e., receivers cannot send two consecutive Early RTCP packets). After sending an Early RTCP packet, the RTCP reporting engine must schedule the transmission time for the next RTCP packet by skipping the next Regular RTCP interval.

Even though the mechanisms proposed in RFC 4585 were not specifically targeted for multimedia synchronization purposes, their application can indeed be very beneficial to enhance the synchronization performance, as discussed in Chapter 9.

As a summary, the stepwise calculation process for the RTCP report interval, using the timing rules specified in RFC 3550 and RFC 4585, is given in Figure 7.3.

7.5.3 Rapid Inter-Stream Synchronization (RFC 6051)

In multimedia streaming services, the inter-stream synchronization delay refers to the time difference between the instant at which a user joins an on-going session, probably involving different media (e.g., audio and video, or when using layered and/or multi-description media) carried in separate streams, and the instant at which these correlated streams can be presented to that user in a synchronized manner.

When using RTP streaming, that delay can greatly increase in certain scenarios, especially in large multicast groups or when Multipoint Conference Units (MCU) are involved in the media delivery process. This increase of delay can be inacceptable and annoying to users, resulting in an overall poor QoE.

The aim of RFC 6051 [Per10b] is to minimize the inter-stream synchronization delay when using RTP/RTCP-based streaming. The motivation is that a receiver cannot synchronize playout of the incoming media streams until compound RTCP packets (RFC 3550), with an SDES packet (including the media source identification) and an SR packet (including timing correlation parameters) are received for all the involved RTP senders in a multimedia session. In most

implementations, media data will not be played out (watched or listened) until inter-stream synchronization is (initially) achieved. If there is no packet loss, this gives an expected delay equal to the average time for receiving the first RTCP packet from the RTP Session with the longest RTCP report interval[23]. This delay is even more problematic if an RTCP SR packet from one of the involved RTP sessions is lost.

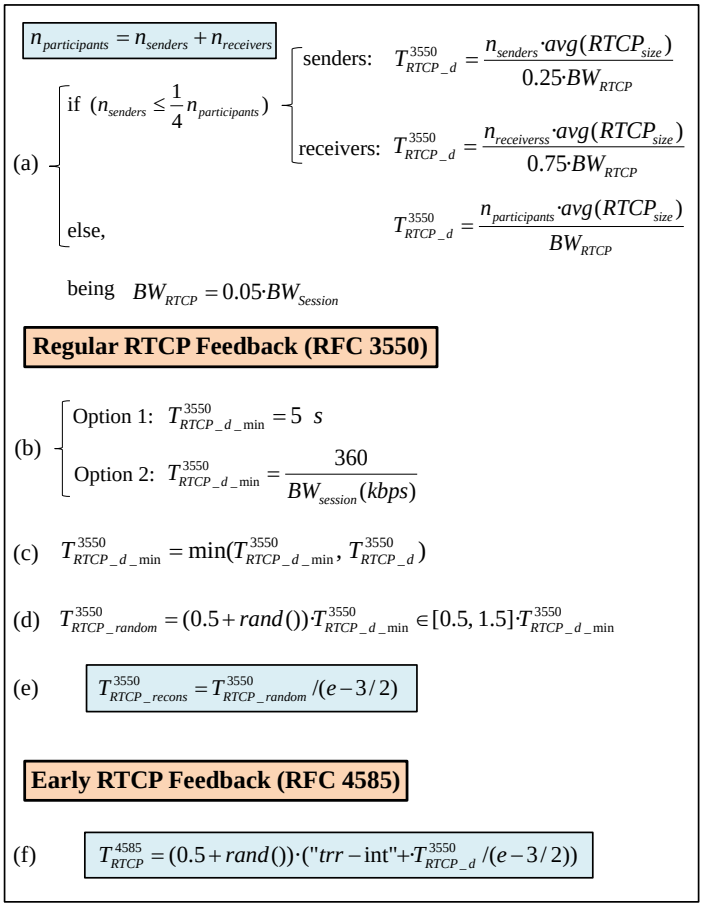


Figure 7.3. Calculation Steps of the RTCP Report Interval.

[23] Note that the inter-stream synchronization delay depends on the specific instant at which a user joins the multimedia session or each RTP session (e.g., the user may first receive the RTCP packets from the RTP session with the longest RTCP interval), as well as on the impact of the randomization processes in all the involved RTP sessions.

In RFC 6051, three backwards compatible extensions to the RTP/RTCP protocols are proposed to reduce the inter-stream synchronization delay. First, the RTCP timing rules are updated to allow SSM senders (RFC 5760) [Ott10a] the immediate transmission of an initial compound RTCP packet upon joining each RTP session in a multimedia session (in parallel with the initial RTP packets). The rationale for not allowing the transmission of immediate RTCP packets to SSM receivers is to avoid feedback implosion in case that many receivers join the session almost simultaneously (which is known as *"flash crowd"* effect). This is clearly not an issue for SSM senders, since there can be at most one sender. Likewise, feedback implosion is a concern for Any Source Multicast (ASM) sessions, so RFC 6051 does not propose changes to the RTCP timing rules in these kinds of multicast environments. Second, a new RTP/AVPF transport layer feedback message (this type of RTCP messages are defined in RFC 4585), called RTCP-SR-REQ, is defined to allow requesting the generation of an Early RTCP SR packet from the media sender. This enables rapid (re-)synchronization in case that an RTCP SR has not been received for a long period (e.g., due to packet loss or in sessions with large RTCP reporting intervals). Likewise, this enables latecomers to achieve inter-stream synchronization as soon as possible upon joining the session. Finally, new RTP header extensions are defined to enable the inclusion of metadata (in particular, NTP-based timestamps) in RTP packets for in-band synchronization, thus avoiding the need for receiving RTCP SR packets before streams can be synchronized. These RTP header extensions do not eliminate the need for RTCP SR messages, but both mechanisms must be used for the synchronization control process. The use of RTCP SR packets for inter-stream synchronization allows backwards compatibility, but also provides higher robustness in the presence of middle boxes (e.g., RTP translators) that might strip RTP header extensions.

An accurate and rapid inter-stream synchronization is especially relevant when using layered, multi-description and multi-view media encodings. This is because all the individual RTP streams need to be synchronized before starting the decoding processes. In these cases, it is useful to insert header extensions into RTP packets corresponding to exactly the same sampling instant in all the involved RTP streams. Accordingly, as all these RTP extensions will have identical NTP-format timestamps, the RTP timestamps for the component streams can be more rapidly and accurately aligned. The frequency of insertion of RTP header extensions must meet a trade-off between the synchronization delay and the added traffic overhead. A recommended solution in RFC 6051 is to insert them at least once per Random Access Point (RAM) of the media.

7.5.4 SDP Modifiers for RTCP Bandwidth (RFC 3556)

In some applications, it may be appropriate to specify the RTCP bandwidth independently of the allocated RTP session bandwidth. Accordingly, RFC 3556

[Cas03] defines two SDP attributes to specify modifiers for the RTCP bandwidth for senders and receivers.

On the one hand, using a separate parameter allows rate-adaptive applications to set an RTCP bandwidth consistent with a "typical" data bandwidth that is lower than the maximum bandwidth specified by the session bandwidth parameter. This allows keeping the RTCP bandwidth under 5% of the session bandwidth when the rate has been adapted downward, e.g. based on the stability of the network conditions. On the other hand, there may be applications that send data at very low rates, but need to exchange quite frequent RTCP packets. These applications may need to specify an RTCP bandwidth higher than 5% of the data bandwidth.

If any of the SDP attributes for the RTCP bandwidth modifiers are omitted, the default value for that parameter is the one specified in the RTP profile in use for the session. RFC 3556 does not impose limits on the values that may be specified for both RTCP bandwidth modifiers. However, the RTP specification and the appropriate RTP profile may specify limits.

7.6 Summary

This Chapter has presented an overview to the RTP/RTCP standard protocols, paying especial attention to their capabilities to enable multimedia synchronization. First, the relevance of such protocols in the current media delivery ecosystem has been discussed. Second, the supported transport-level configurations when using RTP streaming in multimedia sessions have been introduced. After that, the features of such protocols (specified in RFC 3550) to enable intra-media and inter-media synchronization have been described. Finally, the timing rules for exchanging RTCP packets between the participants in RTP sessions have been summarized.

The content of this Chapter is useful to help understanding the next two Chapters, especially our rationale for extending such RTP/RTCP protocols for IDMS purposes and the features of the designed IDMS solution.

Chapter 8

RTP/RTCP-BASED IDMS SOLUTION

8.1 Introduction

This Chapter presents all the different components (i.e., protocols, schemes, algorithms and adjustment techniques) of the RTP/RTCP-based IDMS solution designed in this PhD thesis, as well as the different alternatives that have been developed for several of them.

Although the IDMS solution has been developed by adopting the different IDMS schemes (SMS, DCS and M/S Scheme), the overall IDMS solution is described in this Chapter, because of the various common components when making use of each scheme. However, the particular operational aspects, implementation issues, as well as the extra features that are needed for each control scheme, are highlighted in this Chapter.

8.2 Suitability of RTP/RTCP for IDMS

In this section, our rationale for choosing RTP/RTCP protocols for being extended for IDMS purposes is provided.

After comprehensively studying the features of RTP/RTCP protocols to provide both intra-media and inter-media synchronization (in Chapter 7), their compliance with most of the derived key requirements for IDMS (in Chapter 6) was identified. In particular, the requirements that are inherently met using RTP/RTCP are: R1) RTP packets provide useful metadata for carrying out IDMS (such as sequence numbers, timestamps, PT identifier, source identifier and Marker bit); R7) RTP/RTCP protocols are commonly used in best-effort packet-switched networks, without the need of priority- or reservation-based mechanisms; R8) RTP/RTCP

protocols support (both server-side and client-side) rate adaptive mechanisms; R9) RTP/RTCP protocols are valid for conveying multiple media types (especially audio and video), with different formats; R10) RTP/RTCP protocols support both CBR and VBR encoding mechanisms; R11) RTP/RTCP protocols are supported by both network and end-systems entities; R12 and R13) RTP/RTCP protocols are widely used in a plethora of CoD and live streaming services, such as IPTV, VoIP, and conferencing, all of them requiring IDMS; R14) RTP/RTCP protocols can be used in unicast and in different multicast transmission modes, such as ASM and SSM; R16) RTP/RTCP protocols are standardized within IETF, are under continuous evolution, and have also been adopted by other many standardization bodies, such as ETSI, OIPF, 3GPP and W3C. This helps to ensure inter-operability and widespread support, as well as to promote deployment in real environments.

Furthermore, RFC 3550 allows modifying and/or extending RTP/RTCP protocols to include profile-specific information required by particular purposes (e.g., extensions to existing packets, definition of new RTCP packets, support of new encoding mechanisms, specification of enhanced RTCP reporting rules…). RFC 5968 [Ott10b] provides guidelines that must be followed for extending RTP/RTCP protocols, with the intention of preventing an extension creep that can only harm inter-operability and the future evolution of such protocols at large. Likewise, RFC 3611 [Fri03] allows the definition of new RTCP Extended Report (XR) blocks for exchanging additional QoS metrics regarding media transmission or reception required in specific contexts. Accordingly, the feasibility of extending RTP/RTCP protocols to meet the remaining requirements for IDMS (in particular R2, R3, R4, R5, R6 and R15) was also assessed.

Regarding R2, it was explained in Chapter 6 that RTCP is used as an adaptive and scalable feedback channel between the participants in an RTP session to mainly inform about QoS statistics. To achieve IDMS, the exchange of information about reception and/or playout timing among participants is needed. One possibility in this respect is to report on arrival and/or presentation times for specific RTP packets, which can transport a fragment of one, one, or more than one application-layer MUs (e.g. video frames or audio samples). As this kind of information can be considered a QoS metric (it can reflect the effect of jitter, network load, packet losses, clock deviation, presentation skews, processing delays, etc.), the extension of RTCP becomes a suitable option for carrying out IDMS. Besides, the definition of new RTCP messages to include such information about IDMS will allow continuously monitoring and adjusting the reception and/or playout timings of all the involved Sync Clients during the session's lifetime. Moreover, using RTCP, the optimum transmission rate of feedback messages does not need to be computed (as required by most of the existing IDMS solutions compiled in Chapter 4), since it is dynamically adjusted according to the session membership and available bandwidth. The RTCP timing rules specified in different IETF standards were described in Section 7.5.

Regarding R3 and R4, the RTCP messages to be defined for IDMS can include information about current presentation/playout times of each Sync Client. This will allow synchronizing media streams at the packet level but, at the same time, tackling the IDMS problem above the transport level, as close as possible to the "point of playout" (see Figure 8.1). Accordingly, the end-to-end delay differences between Sync Clients can be compensated for when using RTP/RTCP for IDMS.

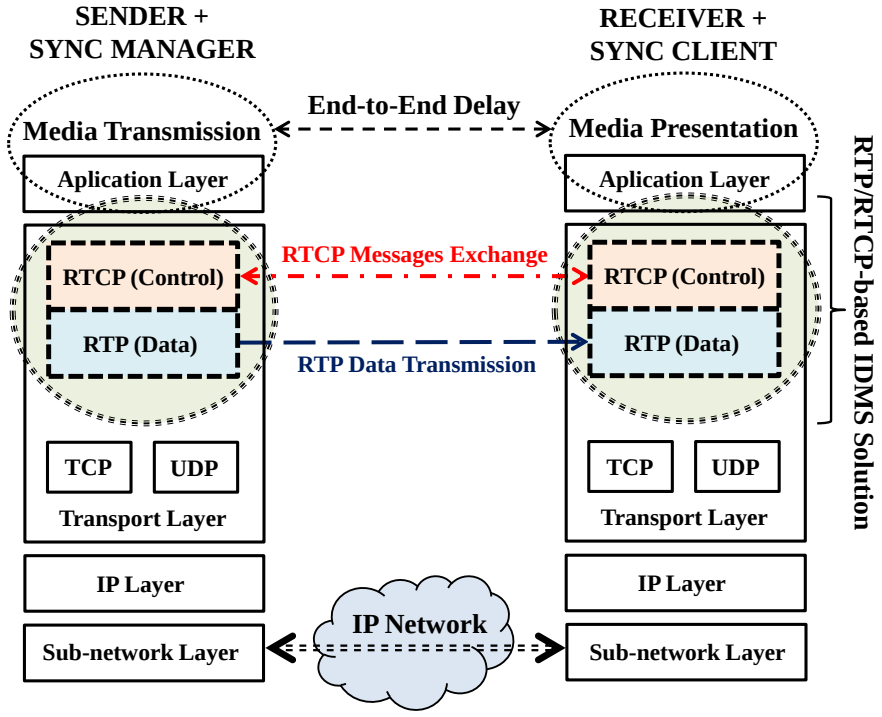


Figure 8.1. RTP-RTCP-based IDMS solution in the TCP/IP Protocol Stack.

Regarding R5, as described in Chapter 6, the use of NTP-based timestamps in RTP streaming allows for accurately aligning the media timing across RTP streams. Such reference timing information can also be used for IDMS. Moreover, further mechanisms can be used to negotiate and inform about the usage of common (or somehow related) wall-clock sources for all the participants in a shared session, as specified in RFC 7273 (summarized in Section 8.5) [Wil14].

Regarding R6, the timestamps included in each RTP packet enable an axis-based synchronization control, while the regular exchange of NTP-based timestamps, both in RTCP SR packets and in header extensions to specific RTP packets, allow for control-based synchronization. Both synchronization methods, which are already available when using RTP/RTCP protocols, can be extended with other mechanisms

to enable dynamic and early IDMS adjustments as a response to either unforeseeable or scheduled events, thus also enabling event-based synchronization control (described in Chapter 9).

Finally, regarding R15, RTP/RTCP protocols can be used in various networked scenarios, with different available resources and conditions (e.g., bandwidth, latency, multicast support…), with variable numbers of participants, and even following different centralized and/or distributed architectural approaches.

Despite the many advantages of using RTP/RTCP for IDMS, the suitability of other standard protocols, such as SIP (RFC 3261) [Ros02], RTSP (RFC 2326) [Sch98], Diameter (RFC 3588) [Cal03], and H.248 (RFC 3525) [Gro03], for being extended for IDMS was also analyzed. SIP and RTSP could be extended with synchronization parameters, but those protocols are not supported by network elements transporting the actual media streams, only by user's terminals. This fact limits the implementation of a network-based IDMS solution [Sto10]. Likewise, RTSP is commonly used for CoD services, but not for live services. Only an RTSP-based IDMS solution would clearly not be sufficient. Diameter and H.248 are protocols used in Next Generation Networks (NGNs) that link the service plane to the transport plane. These protocols could also be extended for IDMS, especially H.248, but, contrarily to the previous protocols, the downside here is that they are not supported by user's terminals, being only applicable for in-network synchronization [Sto10].

Accordingly, RTP/RTCP protocols were selected as the best candidates for tackling the IDMS problem, as their inherent features, in conjunction with their extension capabilities, allow meeting all the identified requirements for IDMS (both the essential and recommended ones).

8.3 Background: Preliminary Version of our RTP/RTCP-based IDMS Solution

This PhD thesis started from a preliminary version of a centralized IDMS solution, based on simple extensions to RTP/RTCP protocols [Bor08], which, in turn, was based on two previous synchronization protocols: *Feedback Protocol* [Ran95] and *Feedback Global Protocol* [Gue01]. The former uses local clocks whereas the latter uses a global time reference. Both are adaptive, valid for multicast and use an M/S Scheme and feedback techniques to exchange synchronization information between sources and receivers.

That earlier version of our RTP/RTCP-based IDMS solution employs an SMS to carry out the synchronization control [Bor08]: the Media Server acts as the Sync Manager, and it considers a Sync Client as the (master) IDMS reference. Once the Sync Clients receive the IDMS setting instructions from the Sync Manager, they perform reactive playout skips/pauses (i.e., aggressive playout adjustments) to

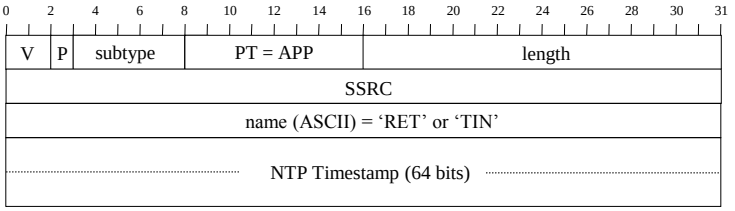
synchronize with the IDMS reference. Likewise, that IDMS solution relies on the availability of a global time reference (e.g., provided by NTP), for all the involved participants in the shared session.

The following RTCP extensions were proposed in [Bor08]. First, RTCP Receiver Reports (RRs) were extended (calling them RTCP RR EXT packets) to include the playout point of each Sync Client. Concretely, the sequence number of the MU being played out and its playout time were included. Second, three new RTCP Application-Defined (APP) packets were defined in order to estimate network delays prior the transmission of MUs and to send playout setting instructions to the Sync Clients. In a later work [Bor09c], an additional field was added to the proposed RTCP RR EXT and APP packets to include the identifier of the logical group to which the sender of such packets belongs (in RTCP RR EXT) or of the group of Sync Clients to which this packet is sent (in RTCP APP ACT). This allows an independent, but concurrent, IDMS control for various groups of Sync Clients. The proposed RTCP messages and their purpose are summarized in Table 8.1, while their format is in Figure 8.2. More details about their specific fields can be found in [Bor08].

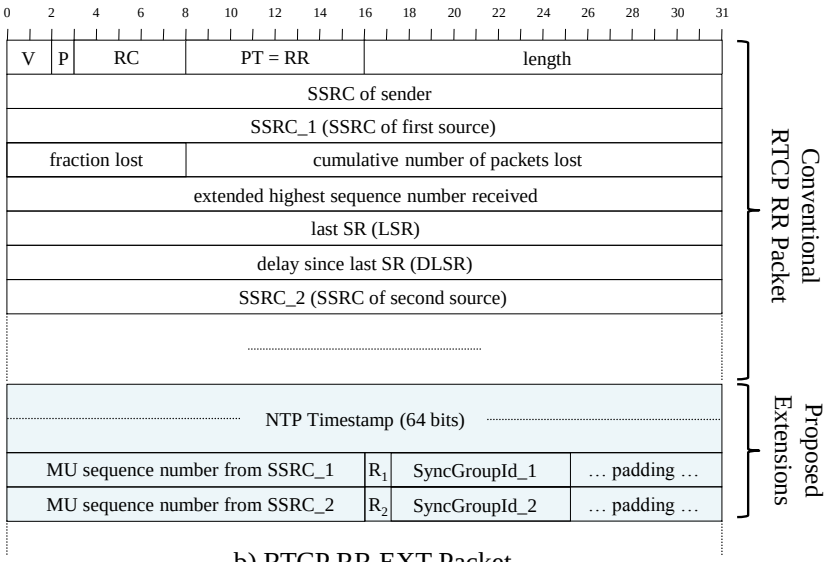
Table 8.1. Proposed RTCP Extensions in the Preliminary version of our IDMS Solution ([Bor08])

RTCP Message	Size (32-bit words)	Purpose
APP RET	5	Sent by the Sync Clients to allow the Sync Manager to estimate the network delay before the transmission of the RTP stream.
APP TIN	5	Sent by the Sync Manager to indicate the global initial playout instant to the Sync Clients
RR EXT	11 (3 extension)	Sent by the Sync Clients to allow the Sync Manager to gather the overall playout information (in each group)
APP ACT	6	Sent by the Sync Manager to indicate the required playout adjustments to the Sync Clients

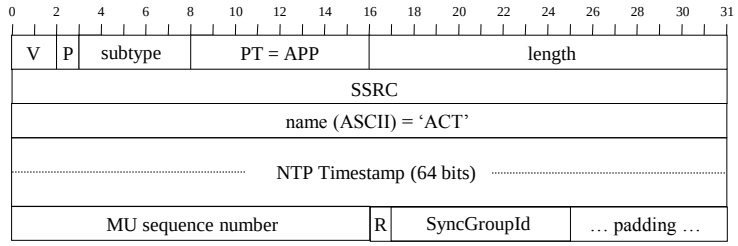
The proposed RTCP extensions for IDMS in [Bor08] were intended to minimize the traffic overhead and to operate within controlled scenarios. Experimental tests in [Bor08] and [Bor09c] proved the satisfactory performance of the earlier solution extensions to achieve IDMS, as well as inter-media synchronization (concretely, audio/video synchronization), for own-designed video sharing or surveillance applications, in real (but controlled) scenarios, between our University Campuses at Polytechnic University of Valencia (UPV, Spain).



a) RTCP APP RET/TIN Packet



b) RTCP RR EXT Packet



c) RTCP APP ACT Packet

Figure 8.2. Format of the proposed RTCP Packets in the Preliminary Version of our IDMS Solution (starting point of this PhD thesis) [Bor08].

However, the deployment of that preliminary version of our RTP/RTCP-based IDMS solution in (uncontrolled) large-scale scenarios, involving third-party infrastructure and communication devices, may present some limitations:

i) Despite that "profile-specific" extensions to RTCP RR are allowed in RFC 3550 for *"giving additional feedback information"* [Sch03] and that they introduce the lowest possible overhead, they may not be backwards compatible with other profiles, as pointed out in RFC 5968. Consequently, they would break the operation and cause inconsistences in some RTP end-systems and middle boxes, lowering the chances of successful deployment. When compatibility is pursued, it is recommended in RFC 5968[24] to define new RTCP XR blocks or RTCP packet types instead.

ii) There is the same problem when considering the inclusion of the IDMS setting instructions in extended RTCP SRs. For that purpose, it seems more reasonable to define a new APP packet. However, APP packets should be used for private (i.e., vendor) specific extensions that do not need to inter-operate with others, or for experimental purposes before specifying new RTCP extensions or registering new RTCP packet types, but the adoption of such packets in standard compliant solutions is inadequate, as specified in RFC 5968.

iii) When including the RTP sequence number to identify the last presented MU in each Sync Client, it is difficult to infer the asynchrony between them if VBR encoding mechanisms are used, which is usually the case in modern multimedia streaming services. This is because the rate of advancement of sequence numbers depends on the temporal and spatial compression methods employed in such encoding mechanisms, as well as on the configured parameters (e.g., GOP size and pattern, quantizer scale...). This means that a variable number of RTP packets may be sent during a specific time interval, which may lead to inaccuracies when comparing the playout timings of the Sync Clients (based on their reported sequence numbers).

Moreover, with the advent of the distributed media consumption paradigm, emerging requirements arose. First, inter-operability between (third-party) implementations and devices needs to be guaranteed when deploying IDMS. Accordingly, standard compliant RTCP extensions need to be specified. Second, some IDMS use cases require very stringent synchronization levels (as discussed in Section 2.4), so highly accurate IDMS solutions must be provided. Third, there is a need of signaling and control mechanisms to negotiate and to inform about key aspects for IDMS (e.g., usage of the RTCP messages for IDMS, groups' establishment, wall-clock sources…). Fourth, as concluded in Section 5.3, the use of SMS is not always the best choice for IDMS, but DCS and M/S Scheme can be more suited than SMS in specific cases. Therefore, the development of such control

[24] It is important to mention that RFC 5968, which provides the guidelines for extending RTP/RTCP protocols, was published later than the design, evaluation, and publication, of that preliminary version of our RTP/RTCP-based IDMS solutions.

schemes will allow efficiently deploying our IDMS solution in a variety of scenarios, according to the targeted requirements or available resources (as discussed in Chapter 5). Moreover, for each control scheme, it is required to explore the suitability of several dynamic strategies for choosing the reference IDMS timing to synchronize with. It is not efficient to tackle a fixed Sync Client as the synchronization reference, because of the variable and unpredictable network and end-system conditions. This also helps avoiding single points of failure. Finally, it has been demonstrated in previous works (e.g., [Su09], [Hss11]) that aggressive playout adjustments lead to a poor QoE. Due to this, the adoption of AMP mechanisms for IDMS is also necessary.

All the above issues reflect a clear need of designing an evolved, extended, adaptive, more accurate and standard compliant IDMS solution to meet the targeted requirements. However, the initial route of extending RTP/RTCP for IDMS is still followed, due to the promising results obtained in [Bor08] and in [Bor09c], the compliance with the derived requirements for IDMS, and the impact that an IDMS solution based on RTP/RTCP would have, as such protocols are widely adopted and supported for streaming media. Furthermore, newer advanced aspects and functionalities to enhance the IDMS performance, for each control scheme in use, have been also devised. The different components (i.e., protocols, schemes, algorithms and adjustment techniques) of such evolved IDMS solution are explained in next sections.

8.4 New Standard RTCP Extensions for IDMS (RFC 7272)

Based on the initial idea in [Bor08], standardization processes were undertaken to provide RTP/RTCP-based technology (i.e., protocols, feedback messages, control mechanisms and architectures) for IDMS under the umbrella of both the ETSI (European Telecommunications Standards Institute) TISPAN (Telecommunications and Internet converged Services and Protocols for Advanced Networking), in [ETSI TS 183 063], and the IETF, in RFC 7272 [Bra14]. Standardization of IDMS started in the ETSI TISPAN, which is a major European-based standardization organization with significant operator involvement. ETSI TISPAN mainly focus on devising new specifications for NGNs and its associated services, working closely together with the 3rd Generation Partnership Project (3GPP). The first release of the ETSI TISPAN standards [ETSI TS 181 061] was focused on regular IPTV services, as well as on CoD services. However, the third release of the ETSI TISPAN standards [ETSI TS 183 063] contains a series of specifications for advanced large-scale IPTV services, including personalization, Social TV and IDMS features. Later on, the IDMS standardization efforts were moved to the IETF Audio Video Transport Core (AVTCORE) Working Group (WG), which is the organization responsible for standardization of newer RTP/RTCP functionalities and updates.

The ETSI TISPAN proposal for IDMS is a dedicated solution for use in large-scale IPTV deployments. However, many other media services may also benefit from IDMS, some of them requiring stricter synchronization levels than IPTV, which are not supported by the ETSI TISPAN specification. Therefore, the goal in our IETF standardization work was to specify a wider-applicable (i.e., valid for different use cases) and more accurate IDMS solution. Although the IDMS specification within the IETF significantly evolves the one specified within the ETSI TISPAN, it was carefully designed to be backwards compatible with this earlier one. Moreover, upon the acceptance of the IETF proposal for IDMS as a standard, in RFC 7272 [Bra14], the ETSI TISPAN proposal was updated to include RFC 7272 as the normative specification for IDMS.

The IDMS standardization efforts were targeted to solve the compatibility constraints of the proposal in [Bor08], by mainly defining standard compliant extensions to RTP/RTCP protocols. The initial steps consisted of choosing the most suited RTP/RTCP extension points, as well as deciding what kind of information about IDMS to include, according to the targeted requirements. RFC 5968 indicates that the definition of *"a new RTCP XR block type is appropriate for transporting new metrics regarding media transmission or reception quality"*. Accordingly, a newly-defined RTCP XR block for IDMS, called "IDMS report", was specified to enable Sync Clients to provide feedback about reception and/or presentation times for specific RTP packets. The IDMS report was registered with Internet Assigned Numbers Authority (IANA), with a Block Type (BT) value of '12'. The IDMS report consists of eight 32-bit words (see Figure 8.3), plus two 32-bit words for its header, and is composed of the default XR headers (RFC 3611), followed by the useful parameters for IDMS, such as: i) the Payload Type (PT) of the media stream this block reports on. This field is needed in case that the Sync Manager is neither the Media Server nor a Sync Client (since it determines the rate of advancement of the RTP timestamps reported by each Sync Client); ii) the SSRC of the source of the media stream this block reports on; iii) the Media Stream Correlation Identifier field, which contains the Identifier of the Synchronization Group (called SyncGroupID) the sender of this report belongs to; iv) the (generation) RTP timestamp (32-bit) belonging to the RTP packet the IDMS report refers to; v) the packet reception time (64-bit NTP-based timestamp); and, optionally, vi) the packet presentation time (32-bit central word of a 64-bit NTP-based timestamp).

In each RTCP report interval, the Sync Clients must report on RTP packet reception times and, optionally, on RTP presentation times. RTP packet arrival times are more accessible to Sync Clients and therefore relatively easier to report on. If the capabilities of the involved end-systems are in some way homogeneous (i.e., with similar buffer settings, and with similar decoding, processing and rendering delays), an acceptable accuracy can be achieved when only reporting on RTP reception times. Nevertheless, in case of variable end-systems' delays, synchronizing on packet arrival times can lead to a loss of accuracy for IDMS.

Accordingly, if all the Sync Clients have the ability to report on, and thus synchronize on, actual playout or presentation times, this will enable high accurate end-to-end synchronization (see Figure 8.1). However, the use of packet presentation times requires the Sync Clients to track RTP packets (transport layer) until their ultimate presentation times (application layer). This can be seen as a form of layer-violation, and some applications or media players could not (easily) link the RTP plane to the application layer, thus implying more difficult implementation requirements. Because of the previous discussion, reporting on packet arrival times is mandatory, but reporting on presentation times is optional when using our RTP/RTCP-based IDMS solution. A *Packet Presented NTP timestamp* (P) flag (1 bit) is used to explicitly indicate if the Sync Clients report on presentation timestamps (set to one) or only on reception times (set to zero).

A more detailed explanation of the fields of the IDMS report can be found in RFC 7272 [Bra14].

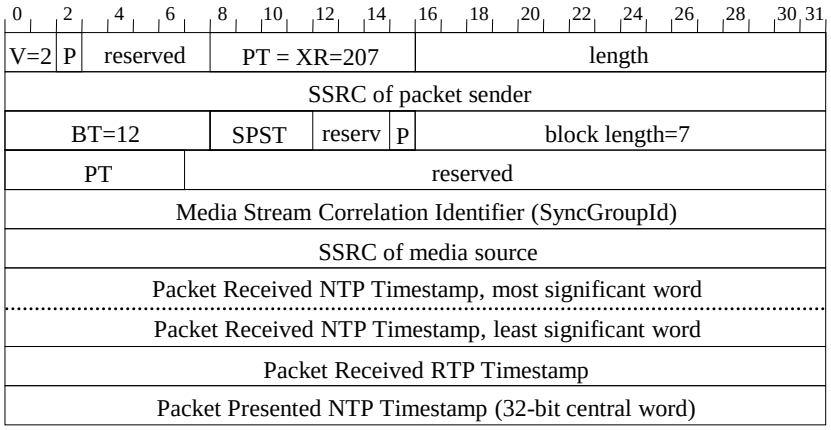


Figure 8.3. RTCP XR Block for IDMS ("IDMS Report").

Apart from reporting and monitoring the IDMS timing of the involved Sync Clients, a control mechanism to notify them the required adjustments to achieve IDMS is also needed, especially if an SMS-based communication model is adopted. RTCP is somewhat less suited for this second purpose, since the transmission of IDMS setting instructions involve some form of application-level control, and it could also be considered as a layer-violation. Moreover, RTCP is not targeted to be a command-based protocol. However, it does make sense to use a single protocol for both reporting and setting purposes.

Originally, the IDMS specification in ETSI TISPAN proposed the use of the IDMS report for both monitoring and setting purposes. In such a case, the Synchronization Packet Sender Type (SPST) field of the IDMS report is used to

indicate the originator of the IDMS report. If the IDMS report is sent by a Sync Client, it will contain its IDMS timing. If the IDMS report is sent by the Sync Manager, it must be interpreted as the reference IDMS timing to synchronize with.

However, according to RFC 5968, RTCP XR blocks must be used only for monitoring, but not for control purposes. For the purpose of IDMS timing control, none of the extension points considered in RFC 5968 is adequate. In this aspect, RFC 5968 states that the only valid reason to create a new RTCP packet type is when the targeted functionality would not be appropriate to be included as part of one of the available packet types or report blocks. As a result, during the IETF standardization process, the consensus was to specify a new RTCP packet type for IDMS settings purposes. It is called "IDMS Settings" packet, and allows the Sync Manager (in case of using SMS) to provide guidance on when to play out the media, as a means of playout hints. The Sync Manager, after collecting the IDMS reports from the Sync Clients, will compute the delay differences among them and, if needed, will send an IDMS Settings packet including a common target playout point (i.e., reference reception and presentation times for a specific RTP timestamp) to which all the Sync Clients belonging to a specific group (identified by the SyncGroupId field) must synchronize. Possible strategies for choosing the reference IDMS timing, for including the IDMS target playout point, and for performing the required adjustments will be discussed in next sections.

Each particular implementation can decide to transmit an IDMS Settings packet in each RTCP report interval, or only when the asynchrony between the Sync Clients exceeds allowable (pre-configured) thresholds. This also applies to the use of the IDMS report for settings purposes in the ETSI TISPAN proposal.

The RTCP IDMS Settings packet consists of seven 32-bit words (see Figure 8.4), plus two 32-bit words for its header, and has been registered with IANA with a Packet Type (PT) value of '211'. It mostly contains the same fields as the IDMS report. However, for achieving highly accurate synchronization, a 64-bit presentation timestamp field has been adopted, instead of the 32-bit field of the IDMS report. This allows higher granularity for those use cases requiring stringent synchronization levels, such as audio beamforming or networked video walls (use cases described in Section 2.4).

A more detailed explanation of the fields of the IDMS Settings packet can be found in RFC 7272 [Bra14].

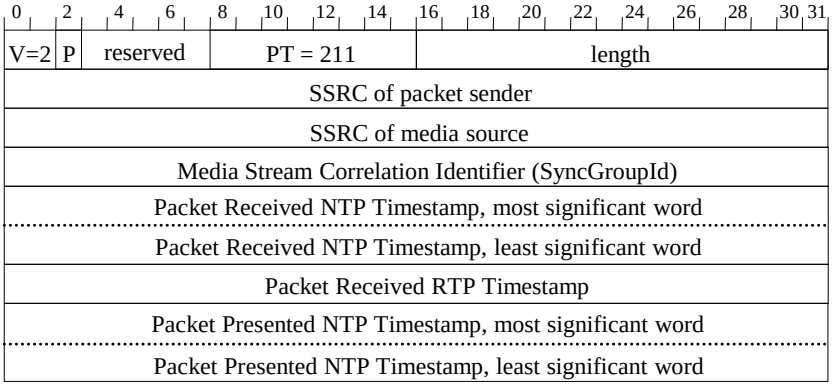


Figure 8.4. RTCP Packet Type for IDMS ("IDMS Settings Packet").

The newly defined RTCP messages for IDMS use (generation) RTP timestamps for identifying specific RTP packets, instead of RTP sequence numbers used in [Bor08]. This helps solving fragmentation issues when application-layer MUs (e.g., video frames) are carried in several RTP packets, since all of them will include the same timestamp. Moreover, a major advantage is that it allows inferring the asynchrony between the involved Sync Clients, even when using VBR coding mechanisms (but with constant MU rate, e.g. a Media Server transmitting with a specific rate of θ=25 frames per second or fps). Moreover, as discussed in next sections, the use of RTP timestamps as the reference for IDMS, instead of RTP sequence numbers, also allows simplifying the operation of the control processes for IDMS, such as the calculation of the network and playout delays for each Sync Client, the comparison among their playout points, and the triggering of playout adjustments.

Apart from the above two RTCP messages for IDMS, a new SDP attribute, called *rtcp-idms*, has been specified in RFC 7272 to inform about their usage. Moreover, the *rtcp-idms* attribute is also used to establish and manage the different groups of Sync Clients (each group with a different SyncGroupId) in a shared media session. This enables meeting one of the targeted goals of this PhD thesis: the co-existence of several groups of Sync Clients in a single IDMS-enabled session, while allowing an independent, but concurrent, synchronization of their playout processes. The reason for that is because, although media sessions can involve a large number of users (e.g., a broadcasted TV event), users mostly communicate and interact within small-scale groups (e.g., with family members or friends) [Hua11]. Therefore, it is interesting and beneficial to separately handle the synchronization processes for each individual group.

A group-based synchronization policy was already proposed in [Bor09c]. However, the Sync Clients of each group, called *clusters* in [Bor09c], were placed

in the same local environment. Besides, an 8-bit field was used for allocating the group identifier in [Bor09c] (see the format of the RTCP messages in Figure 8.2), will allows up to 2^8=256 different groups. In the new standard RTP/RTCP-based IDMS solution (RFC 7272), a 32-bit field is used for that, which enables higher scalability and minimizes the probability of collision (i.e., the situation in which two groups choose the same identifier). Moreover, the process of randomly generating identifiers with 32-bit fields is frequently adopted in numerous telecommunication protocols and systems (e.g., the generation of SSRC identifiers in RFC 3550).

Various strategies for grouping the Sync Clients can be used. For example, Sync Clients can be grouped based on semantic or logical information (e.g., friends, family members, ad-hoc groups…), on geographic information (e.g., same city, country…), or even according to their QoS levels (e.g., delay, jitter, percentage of lost packets, available bandwidth...). Moreover, situations in which a specific Sync Client simultaneously belongs to various logical groups could also be supported (i.e., a Sync Client is the linking ping between two non-overlapping groups). For instance, it may be the case in which a user is watching a single TV program simultaneously with different groups (e.g., with family members and with friends).

8.5 Clock Synchronization Mechanisms for IDMS

Most IDMS solutions do require wall-clock synchronization between the involved sync entities, especially those ones exchanging feedback information about IDMS timing (summarized in Table 4.1). Wall-clock synchronization is necessary to calculate one-way delays between sync entities as well as to correlate the IDMS reports from the Sync Clients.

RTP/RTCP protocols use NTP-based (64-bit) timestamps to facilitate multimedia synchronization and to provide a useful means for estimating delays and other statistical parameters. Such timestamps are retrieved by reading a local clock, which may be synchronized to another (probably external) clock source, or may even be un-synchronized.

If the involved participants in a media session employ different un-coordinated wall-clock sources, it is rather difficult to provide accurate estimations of the targeted metrics. Without a common reference clock (providing a global time reference), a specific Sync Client can align RTP streams from the same Media Server using relative timing. However, tight synchronization between two or more different Sync Clients or between two or more Media Servers becomes more difficult. Therefore, the availability of synchronized clocks for the involved sync entities in an IDMS-enabled session is necessary for an accurate interpretation and alignment of NTP-based timestamps. In this context, two or more local clocks are assumed to be perfectly synchronized when they produce timestamps for a given media event in a consistent way, as if they came from the same clock.

The most widely used mechanism for clock synchronization is NTP, specified in RFC 5905 [Mil10]. NTP can provide an acceptable synchronization accuracy in several use cases. A typical example is when synchronizing audio and video (i.e., lip-sync), as in such cases synchronization levels of a few tens of milliseconds are typically sufficient (even though high quality video and high frame rates may require more stringent synchronization levels).

Previous studies focused on determining the performance of NTP for clock synchronization (e.g., [Mil90], [Mrt06] and [Vai11a]). The study in [Mil90] (conducted in 1989) measured the accuracy provided by NTP over 100000 nodes distributed across the world. It was found that the majority (> 50 %) of the clocks in the NTP network where within 10 ms of each other. In other study [Mrt06] (conducted in 2006), it was shown that over 95 % of the nodes in a NTP network were within 128 ms of each other. More recently (in 2011), the research in [Vai11a] aimed at determining if the latest evolutions of Internet and NTP technologies resulted in a better performance of clock synchronization compared to the previous studies. On the one hand, it was shown that any two nodes on the Internet were synchronized within bounds lower than 100 ms with a probability close to 90 %, and that approximately 60 % of the involved nodes were within 20 ms of each other. Such results denote that, using NTP, the nodes connected via the current-day Internet infrastructure can be synchronized to reasonably accurate levels, provided that such nodes are properly administered. Obviously, enterprise or managed networks, can perform much better. Likewise, in local and controlled environments, a local NTP server could be set up to improve the clock synchronization accuracy.

The preliminary version of our RTP/RTCP-based IDMS solution [Bor08] was based on the use of NTP for wall-clock synchronization. Besides, the ETSI TISPAN specification for IDMS [ETSI TS 183 063] also poses the requirement of the use of NTP for wall-clock synchronization. The use of NTP in a managed IPTV deployment is a suitable option and may provide acceptable synchronization levels, since the operator can also provide the NTP servers. However, in the Internet environment, although NTP allows achieving high synchronization levels, its use may lead to less accurate synchronization. One reason can be the use of different NTP servers (with clock offsets between them) by different Sync Clients. Besides, NTP servers are not always set up correctly, and may provide wrong clock timing references. Moreover, clock deviations within the Sync Clients will result in a loss of synchronization accuracy. As a guideline to deal with clock deviation issues, the Sync Clients should synchronize their clocks at the beginning of an IDMS-enabled session. If high synchronization accuracy is pursued, the clocks of different Sync Clients should not drift beyond the accuracy required for the synchronization mechanism. In practice, this can mean that the Sync Clients need to synchronize their clocks repeatedly during an IDMS-enabled session.

Even considering the previous issues, the use of NTP cannot provide the stringent synchronization levels required in some IDMS use cases (enumerated in Section

2.4). For instance, if the goal is to synchronize several audio streams (e.g., different channels in a surround-sound system), then the synchronization requirements become much stricter, making the use of NTP inappropriate. Moreover, it could be possible that not all the involved sync entities support NTP, but support other technologies for clock synchronization. Therefore, the use of other (more accurate) clock synchronization mechanisms, like network protocols (e.g., Precision Time Protocol or PTP [IEEE 1588]) or radio clocks (e.g., GPS clocks) must also be enabled for multimedia synchronization. An overview of different alternatives for clock synchronization, such as NTP, PTP or GPS, including their precision and applicability, is given in [Vai11b].

The designed IDMS solution in this PhD thesis can take advantage of the clock source negotiation and signaling mechanisms specified in RFC 7273 [Wil14]. Although such mechanisms are not a contribution of this PhD thesis, they were derived from the initial versions of our IDMS specification within the IETF, and can be fully integrated with our IDMS solution. This provides a high impact and significant advantages over the preliminary version of the IDMS solution in [Bor08]. That is the reason why such mechanisms are explained in this Chapter.

In particular, RFC 7273 specifies an SDP attribute, called *clocksource*, to allow participants to declare if they support clock synchronization, which clock sources they support, which source is currently being used (by providing its address or identification parameters), and if that clock source is *"traceable"*[25]. Other relevant information, such as the last time the participant synchronized with this clock source and the synchronization frequency was also considered for its inclusion. Such information can be used as an indication of clock synchronization accuracy and allows the involved sync entities in an IDMS-enabled session to negotiate the selection and use of a common, somehow related, or, at least known, clock source (from which NTP-based timestamps will be derived). The specification of the reference clock source via SDP can be given at the session, media or source levels. Definitions and examples of each situation are given in RFC 7273.

Currently, the defined clock sources include local (i.e., no support for synchronization exists)[26], NTP, GPS, GAL (GALILEO), GLONASS (Global

[25] A clock is considered to provide traceable time if it can be proven to be synchronized to International Atomic Time (TAI). Timestamps from clocks obtained from traceable time sources can be directly compared, even if these clocks are synchronized to different sources or via different mechanisms. Coordinated Universal Time (UTC) is a time standard synchronized to TAI, so UTC clocks (e.g., NTP) can provide traceable times. For example, if a sender informs it is using a "traceable" clock (e.g., provided by NTP or PTP), a receiver could use GPS as a reference clock, since GPS is also a source of traceable time.

[26] RFC 3550 allows senders and receivers to either use a local wall-clock reference for their NTP-based timestamps or to supply no timestamps at all (by setting the timestamp field to 0). In such cases, the clocks are identified as "local" and can only be assumed to be equivalent to clocks originated from the same device.

Navigation Satellite System) and PTP. It is important to note that this list is extendable, so other clock synchronization technologies can be added and registered with IANA in the future.

If the clock sources, or their parameters, change during the session's lifetime, SIP protocol can be used to inform about such updated information.

Applications performing IDMS may or may not be able to choose a synchronization mechanism for the system clock, because this may be a system-wide setting which the application cannot change. How applications deal with this is up to the implementation. Alternatives include: i) the application might control the system clock; ii) the application might use a separate application-level clock; or even iii) the application might use a separate clock only for the IDMS-enabled session.

8.6 Architectures and Functional Entities for IDMS

This section describes the architectural approaches and the involved functional entities in both the ETSI TISPAN and the IETF specifications for IDMS.

The ETSI TISPAN standards specify two architectures for IPTV services, both of them supporting the IDMS functionality. The first one is based on IP Multimedia Subsystem (IMS), using SIP for session control, while the second one, called *Integrated IPTV subsystem*, is mainly HTTP-based. This section only focus on the IMS-based IPTV specification, even though both architectures are similar in many aspects.

The architecture for IMS-based IPTV services is described in the second release of ETSI TISPAN specifications [ETSI TS 182 027], from which the ETSI TISPAN proposal for IDMS was designed. The main functional entities and reference points are shown in Figure 8.5.

The ETSI TISPAN proposal for IDMS uses the concept of synchronization sessions, which requires the introduction of two new functional entities (a single Sync Manager and multiple Sync Clients) and one new reference point (called *Sync*) between such functional entities, as shown in Figure 8.6.a. The RTCP messages for IDMS are exchanged between the involved functional entities through this *Sync* reference point.

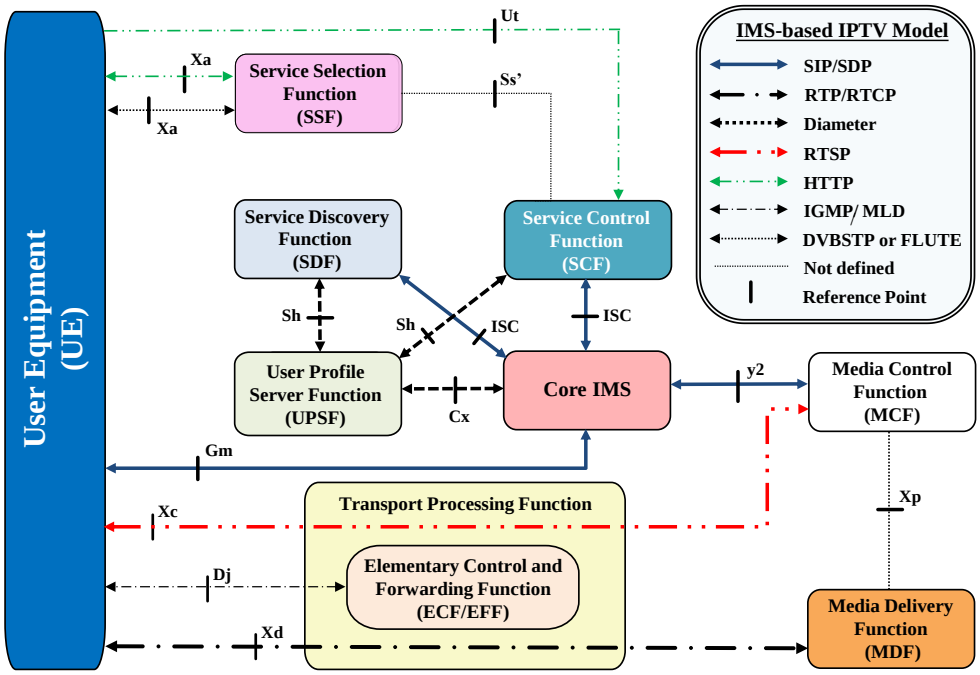


Figure 8.5. ETSI TISPAN functional entities and reference points in the IMS-based IPTV architecture.

Likewise, ETSI TISPAN considers various mappings of the IDMS functional architecture onto the entities in the IPTV architecture. This is one of the main advantages of designing a functional architecture: various implementations are possible with a single specification. One mapping, aimed at small-scale deployments, is based on the use of SMS: the Sync Manager is implemented in the network and the Sync Clients are co-located with the UE (i.e., terminal-based IDMS). The ETSI TISPAN proposal for IDMS defines the Sync Manager as a functional entity separated from the Media Distribution Function (MDF), which is the ETSI term for Media Server, although there is no restriction for co-locating the Sync Manager functionality within the same entity as the Media Server. Another possibility is to use a peer-to-peer communication channel between the involved Sync Clients (co-located with the UEs) to exchange the IDMS messages, which represents the adoption of a DCS. Finally, another mapping, aimed at large scale deployments, consists of placing the Sync Clients within edge nodes of the transport network (i.e., network-based IDMS). An edge node can be, for example, a DSLAM (Digital Subscriber Line Access Multiplexer) or a CMTS (Cable Modem Termination System). Furthermore, at a higher level (e.g., in the core network), a Sync Manager must be used to control the IDMS timing of the Sync Clients.

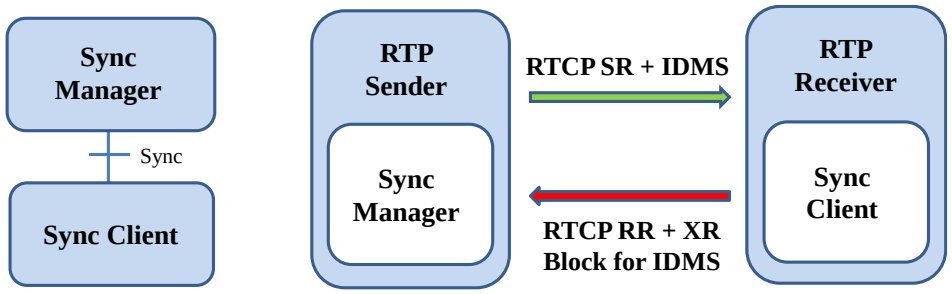


Figure 8.6. Functional Entities and Reference Point for IDMS.

The ETSI TISPAN proposal also includes the protocols for setup, maintenance and teardown of IDMS-enabled sessions. Either on-going regular media sessions can be converted to IDMS-enabled sessions, or new media sessions can be set up directly with the IDMS functionality. If the Sync Clients are located within edge nodes, they need to be configured beforehand with regard to IDMS (since they are not involved in the media session). If the Sync Clients are located in the UEs, the IDMS-enabled sessions are setup using SIP and SDP, using the G_m and ISC reference points (see Figure 8.5), for broadcast services, or using a combination of SIP and RTSP, also using SDP, for CoD services and Personal Video Recorder (PVR) sessions. The SDP media description includes the following IDMS-specific items:

- The address of the Sync Manager. This is allocated by the Service Control Functions (SCFs). Typically, a single Sync Manager will be used in an IDMS-enabled session. However, the setup of various Sync Managers, at either the same or different hierarchical levels, is also allowed in a single IDMS-enabled session.
- A SyncGroupId, which has a similar function as a *conference-ID* in conference calls, where each user has to enter the same *conference-ID* to become part of the same conference call.
- In case of CoD, the SSRC of the RTP media stream. It is necessary because in unicast sessions each RTP stream has a different SSRC identifier. Therefore, the SSRC identifier of each RTP stream is needed to correlate the RTCP messages from the multiple Sync Clients.

From the viewpoint of end users, an IDMS-enabled session can be ended in two ways: i) by ending the entire media session, obviously including any synchronization part, and ii) by reverting the IDMS-enabled session to a regular media session. In order to revert to a media session, a SIP *re-invite* method is sent,

containing an exact duplicate of the session description, but omitting the IDMS-specific parameters. Likewise, if only one Sync Client remains in an IDMS-enabled session, the Sync Manager will terminate that session by sending a similar *re-INVITE* to that Sync Client. Using SIP in this manner allows for flexible setup of IDMS-enabled sessions, not only for CoD services, but also for broadcast services, such as linear TV (e.g., various groups of viewers sharing a TV experience can co-exist, even for the same TV broadcast).

Similarly to other recent application-layer service capabilities for IPTV, such as retransmission or Forward Error Correction (FEC) technologies, the IDMS solution specified by ETSI TISPAN can be implemented as an add-on in existing IPTV deployments

The proposed architecture for IDMS in the IETF specification (RFC 7272) has been simplified. The ETSI TISPAN proposal for IDMS is meant to be very scalable, and thus the synchronization functions have been specified as functions separated from the MDF (i.e., the RTP Sender) and the UE (i.e., RTP Receiver). In the IETF specification, the Sync Client functionality is implemented as part of an RTP Receiver, and the Sync Manager functionality is implemented as part of the RTP Sender (see Figure 8.6.b). Optionally, the Sync Manager can also be co-located with an RTP receiver, as discussed in Section 3.4.

8.7 Exchange of IDMS Messages in each Control Scheme

This Section describes the exchange of RTCP messages for IDMS when using each one of the deployed control schemes (SMS, DCS and M/S Scheme).

During an RTP session, after the Media Server starts sending RTP packets (encapsulating data MUs), each distributed *i-th* Sync Client regularly sends RTCP RR packets to inform about QoS metrics. Additionally, when using SMS and DCS, each *i-th* Sync Client must also send an IDMS report, in each compound RTCP packet, including its local playout point: i) the original RTP timestamp of the MUs being played at that moment (t'_i); ii) its reception time (r_i); iii) optionally, its presentation time (p_i); and iv) the identifier of the *k-th* group the Sync Client belongs to. When using M/S Scheme, only the master Sync Client will send the IDMS report (typically, in a multicast way).

Accordingly, when using SMS, if unicast is used, the IDMS reports will be only received by the Sync Manager. When using DCS and M/S Scheme, if multicast is used, the IDMS reports will be received by all the sync entities involved in the multicast session, including the Sync Clients. However, if various groups of Sync Clients are active in an IDMS-enabled session, each Sync Client must only register and process the information of the incoming IDMS reports from all the other Sync Clients belonging to the same group/s, despite it may receive the IDMS reports from

all the groups in the (multicast) session. This process when using DCS for IDMS is shown in Figure 8.7.

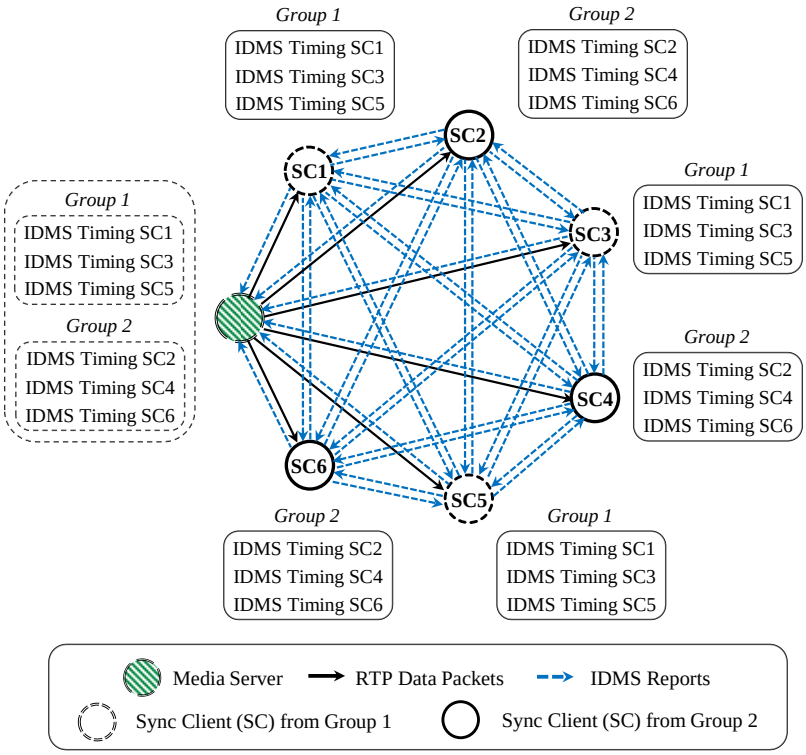


Figure 8.7. Group-based Operation of DCS for IDMS.

This way, once the overall information about IDMS in each *k-th* group has been collected, the playout time discrepancy (i.e., the asynchrony) between the Sync Clients belonging to that group can be computed.

When SMS or DCS are employed, the Sync Manager or each Sync Client, respectively, must compare the local playout (or end-to-end) delays of each *i-th* Sync Client belonging to that *k-th* group, d_i^k. It can be calculated as the time difference between the presentation time of the current MU being played out by that Sync Client (i.e., the one reported in the IDMS report) and the RTP generation timestamp[27] of that MU (more accurately, of one of the RTP packets encapsulating that MU):

[27] For that purpose, the generation RTP Timestamp (32-bits), t'_i, must be mapped to its associated NTP-based timestamp (64-bits): $t'_i \rightarrow t_i$. If the sync entity calculating the playout delay is not the Media

$$d_i^k = (p_i^k - t'^k_i) \qquad \textbf{Eq. 8.1}$$

The maximum asynchrony in each *k-th* group (Δ^k_{max}) will be given by the difference between the most lagged and the most advanced playout points of all the active Sync Clients in that group (G_k):

$$d^k_{\max/\min} = \max/\min\{d_i^k, \forall i \in G_k\} \qquad \textbf{Eq. 8.2}$$

$$\Delta^k_{\max} = (d^k_{\max} - d^k_{\min}) \qquad \textbf{Eq. 8.3}$$

If M/S Scheme is employed, each *i-th* slave Sync Client must calculate the asynchrony with the master Sync Client every time an IDMS report from it is received:

$$\Delta^k_{\max} = (d_i^k - d^k_{master}) \qquad \textbf{Eq. 8.4}$$

The timing diagrams for the RTCP messages exchange in our RTCP-based IDMS solution, when using SMS and DCS, are illustrated in Figures 8.8.a and 8.8.b, respectively. The timing diagram for M/S Scheme has not been illustrated because it is similar to the one when using DCS, but in such a case only the master Sync Client sends IDMS reports.

Server, it can easily be done by using the mapping time information included in the RTCP SRs sent by the Media Server (RFC 3550).

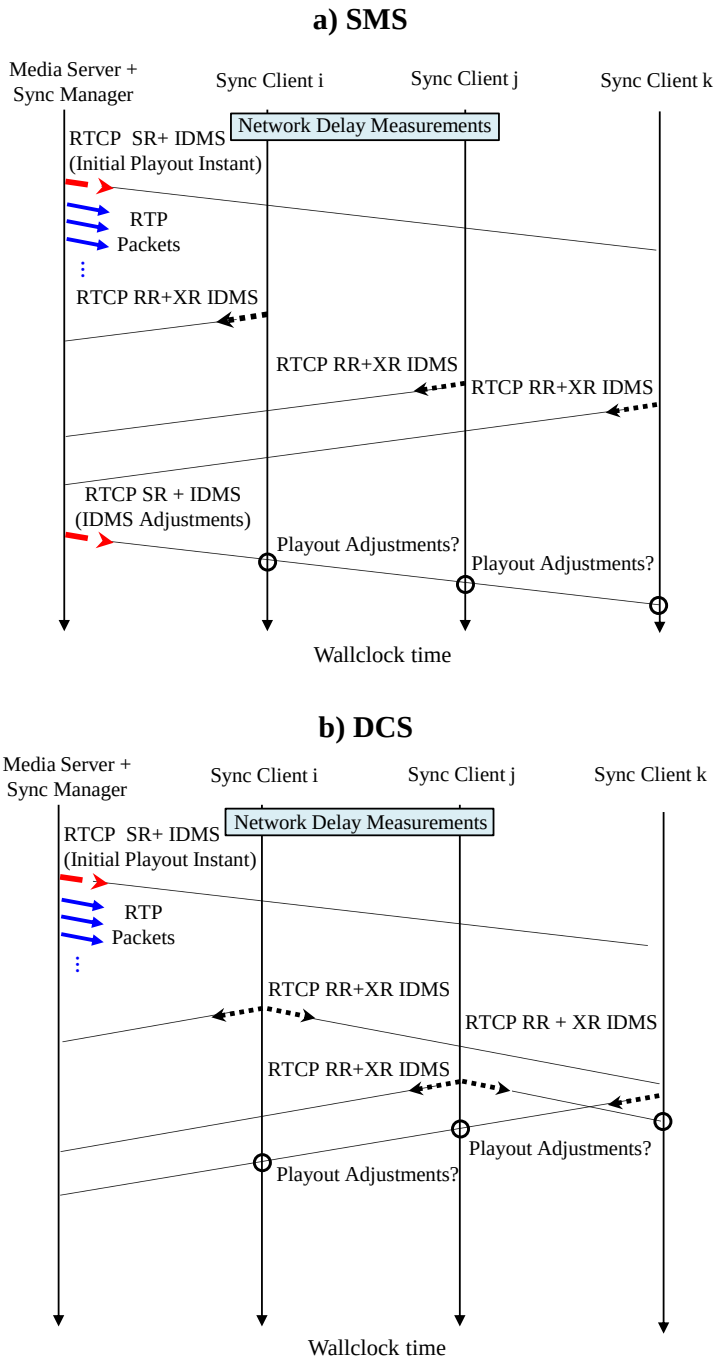


Figure 8.8. RTCP Messages Exchange for IDMS.

8.8 Master Reference Selection Policies

In an IDMS-enabled session, once the IDMS reports from all the Sync Clients in a specific *k-th* group have been gathered, if the detected asynchrony exceeds an allowable threshold τ_{max} (i.e., $\Delta^{k}_{max} \geq \tau_{max}$), reactive adjustment techniques must be triggered to restore the synchronicity. Accordingly, the first decision consists of selecting the master reference playout point to synchronize with.

A commonly used option is to select a fixed Sync Client as the IDMS reference throughout the duration of the session (as done in the evaluation in [Bor08]). Another extended method is to enforce the largest estimated delay to all the involved Sync Clients (e.g., as done in [Mau04] and [Vai11a]). Moreover, the work in [Has06] examined the impact of selecting the most lagged and most advanced playout points as the IDMS reference. However, the assessment of other methods for selecting the IDMS reference was left for further study in that work.

Consequently, one of the goals of this PhD thesis was to analyze, for each IDMS control scheme in use, the feasibility and suitability of various dynamic policies for choosing the master reference playout point to synchronize with. This selection may influence the overall quality of the media session, as it may have an impact on various key aspects, such as the synchronization effectiveness, interactivity, fairness, buffer fullness levels, frequency and magnitudes of the playout adjustments, etc. Moreover, the selection of a specific master selection policy may depend on the specific features of the networked scenario (e.g., bandwidth availability, delays…) and of the involved Sync Clients, as well as on the application requirements.

With the newly defined RTCP messages for IDMS, the processes of comparing the IDMS timing of the distributed Sync Clients belonging to each group (see Equations 8.1, 8.2 and 8.3), and determining the reference IDMS timing to synchronize with, are more accurate and simpler compared with the IDMS solution in [Bor08], because no translation from RTP sequence numbers to timestamps is needed.

Using M/S Scheme, the selection of the IDMS reference is implicit, since it is given by the timing information included in the IDMS reports from the master Sync Client.

Using SMS, the Sync Manager can employ several dynamic policies to select the master reference playout point to synchronize with. Possible policies include: i) synchronization to the slowest Sync Client; ii) synchronization to the fastest Sync Client; iii) synchronization to the mean playout point; and iv) synchronization to the Media Server nominal rate. Additionally, some variations to the above policies could be employed to account for variable network conditions (e.g., adding some extra delay to smooth out extreme jitter or congestion situations).

The first strategy consists of selecting the playout timing of the slowest (i.e., the most lagged) Sync Client in each *k-th* group as the IDMS reference, which is the one with the largest playout delay (i.e., $d^k_{IDMS} = d^k_{master} = d^k_{max}$). Using this method there will not be any skips in the Sync Clients' playout processes, avoiding the subsequent discontinuities, since (faster) slave Sync Clients will be forced to pause their playout processes (waiting for the slowest one). This policy is suitable for multimedia applications with flexible delay requirements, and it could enable the inclusion of interactive error recovery techniques through retransmission requests. Besides, in distributed scenarios where users compete with each other (e.g., battle MOG, or networked quiz shows), this policy is appropriate to guarantee fairness between them. However, if the playout process of the master Sync Client is extremely lagged (e.g., due to any problem, such as network congestion or end-system overload), the use of this policy could result in the progressive filling of the playout buffers of all the Sync Clients, which could eventually overflow. This way, loss of real time sensation would be noticed, affecting the overall QoE. To avoid such situations, additional adjustment techniques, such as buffer fullness monitoring and control, should be deployed (not considered in this PhD thesis).

The second method is the opposite of the previous one, and consists of selecting the playout timing of the fastest (i.e., the most advanced) Sync Client in each *k-th* group, which is the one with the lowest playout delay (i.e., $d^k_{IDMS} = d^k_{master} = d^k_{min}$), as the IDMS reference. As an example, in collaborative scenarios, the efficiency of the overall work may be improved by adjusting the lagged playout timings to the earliest one. Nevertheless, if there are slow Sync Clients under bad conditions (e.g., long network or processing delays), they could be constantly skipping MUs to achieve IDMS, and the continuity of their playout processes could be seriously affected. In such a case, if the playout process of the master Sync Client is extremely advanced, the playout buffers of all the Sync Clients in the same group may suffer underflow (i.e., a progressive emptying of their buffer occupancy) as the media session advances in time. Hence, additional adaptive buffering and control techniques should also be included. This policy could not be applied in live streams, such as linear TV, because of the inability to speed up a live stream, unless large buffering delays are employed at the beginning of the session. However, this results in a significant increase of the latency, which is obviously undesirable, especially considering that channel changing delays (i.e. zapping time) would increase significantly because of this.

The above policies are dynamic processes, since the master and slave roles of the Sync Clients can be exchanged during the session's lifetime, depending on several uncontrollable network and end-system factors, allowing M/S switching techniques [Bor09a].

Another solution for selecting the IDMS reference is to define a virtual playout point (i.e., a fictitious Sync Client), obtained as the mean point of all the gathered playout processes in each *k-th* group (i.e., $d^k_{IDMS} = d^k_{master} = d^k_{mean}$). Using this

method, the playout processes of the Sync Clients will be more continuous and smoother since the number and the magnitudes of the IDMS adjustments will be lower than in the previous ones. However, its use does not guarantee playout buffer overflow or underflow avoidance, because the playout rate imperfections and end-systems situations (e.g., bandwidth availability, network load, CPU congestion...) are unpredictable (i.e., there could be a higher/smaller proportion of fast Sync Clients than slow Sync Clients, with different deviation values). As in the above policies, further adaptive techniques to control the buffers occupancy should be deployed.

Using DCS, each Sync Client must locally, and independently of the other Sync Clients, decide the master reference to synchronize with by using one of the three above discussed master selection policies.

Additionally, a fourth strategy can be adopted, but only when using SMS and when the Sync Manager and the Media Server are co-located. It consists of the synchronization to the Media Server nominal rate. In such a policy, the Sync Manager will act as a virtual master Sync Client with an ideal target playout timing, which is defined as the ideal playout timing when there is neither network nor end-systems' delay variability. Therefore, the maximum asynchrony will be calculated taking into consideration the playout point of this virtual ideal Sync Client as another Sync Client in each group of the IDMS-enabled session. Using this policy, if network conditions are reasonably stable and within allowable limits, underflow and overflow situations will be avoided (assuming that accurate clock synchronization mechanisms are available to all the involved sync entities). This is because the playout states of the deviated Sync Clients in each group will be adjusted to this ideal playout point every time an asynchrony situation is detected. Furthermore, this technique is beneficial for accurate Sync Clients because the smaller the deviations are in their playout states the smaller adjustments would be needed to achieve IDMS.

In all the previous policies, the reporting of an erroneous playout point by a Sync Client, either accidental or malicious, may lead to undesired behavior. According to the adopted model, extremely advanced or lagged playout points will produce frequent and/or high adjustments on the Sync Clients' playout processes, with the subsequent significant loss of real-time or continuity perception. Therefore, in any implementation, with the aim of avoiding faulty behavior, it would be advisable that the Sync Manager in SMS, or the distributed Sync Clients in DCS and M/S Scheme, consider inconsistent playout information (exceeding configured limits) as a malfunction service and reject that information in the calculation of the IDMS target playout point.

Each of the discussed policies are somehow based on choosing (and enforcing) a common end-to-end delay for all the involved Sync Clients in each group. It is well-known that in multimedia systems the end-to-end delay must be kept within allowable ranges during the session's lifetime to avoid buffer underflow and

overflow situations, and to prevent from loss of interactivity. Therefore, as previously discussed, extremely advanced or lagged playout points must not be selected as the IDMS reference. Moreover, an important benefit of reporting on both RTP packet reception times and presentation times is that it allows determining if the origin of a high/low end-to-end delay reported by a Sync Client is due to a high/low network delay or to a high/low end-system delay (i.e., $d^k_i \approx l^k_i + b^k_i$, where l^k_i is the network delay, and b^k_i is the buffering delay, which is mostly equivalent to the end-system delay, for the *i-th* Sync Client belonging to the *k-th* group). For example, a Sync Client with an extremely low network or end-system delay must not be chosen as the IDMS reference because it may force underflow situations, and therefore playout discontinuities, to all the involved Sync Clients. Similarly, a Sync Client with an extremely high network or end-system delay must not be chosen as the IDMS reference because it may force overflow situations to all the involved Sync Clients. Therefore, both the network delays and the end-system delays, and consequently the end-to-end delays, of the Sync Clients must be within allowable limits in order to select such Sync Clients as the IDMS references, as shown in Figure 8.9. The values of such specific upper and lower bounds strongly depend on the application and scenario at hand.

Likewise, if the IDMS timings reported by specific Sync Clients exceed upper or lower thresholds, e.g., due to extreme network or end-systems congestion, additional synchronization techniques should be adopted to dynamically adjust the playout processes of such Sync Clients.

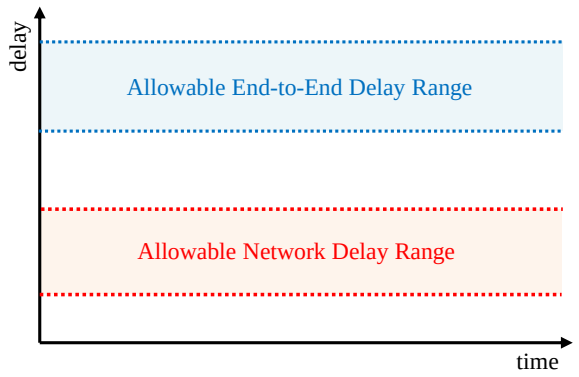


Figure 8.9. Allowable Network and End-System Delay Limits.

8.9 IDMS Target Playout Point and Asynchrony Calculation

In this section, the calculation processes of the target playout point for IDMS in each one of the developed schemes, and of the asynchrony with the selected IDMS reference, are described.

When using SMS, if an out-of-sync situation (i.e., an asynchrony exceeding allowable limits) in a specific *k-th* group is detected, the Sync Manager will calculate a target playout point for IDMS considering the output timing of the selected IDMS reference in that group, following one of the master selection policies presented in the previous section. Then, it will send a new RTCP IDMS Settings packet, including a reference RTP timestamp (t'^{k}_{IDMS}) and its targeted reception (i.e., $r^{k}_{IDMS}=t'^{k}_{IDMS} + l^{k}_{IDMS}$) and playout (i.e., $p^{k}_{IDMS}=t'^{k}_{IDMS}+d^{k}_{IDMS}$) times according to the timing of the selected IDMS reference. This packet can either reflect/forward the timing information of the IDMS report sent by the selected master Sync Client in that group or even include playout hints for a specific (recent or even future) sent MU (included in one or several RTP packets, and identified by its generation timestamp), by inferring the timing evolution of the master Sync Client. In the latter case, the RTCP IDMS Settings packet will refer to the first RTP packet containing a specific RTP timestamp.

If the IDMS Settings refers to a future RTP packet, higher performance in terms of coherence can be achieved, because it will guarantee the simultaneous synchronization of all the Sync Clients to the same IDMS target playout point. Therefore, this is the employed method when using SMS in the proposed IDMS solution.

Let us consider the case that the *i-th* Sync Client belonging to the *k-th* group is playing a specific MU, which is identified by its RTP generation timestamp (t'^{k}_{i}) and by a given *n-th* MU sequence number ($MU_{n,i}^{k}$), at $p_{n,i}^{k}$ instant (local playout point). That Sync Client would consume the successive MUs with a (possibly deviated) [28] playout rate of $\mu^{k}_{n,i}$ MU/s. So, using SMS, it would play out the MU'^{k}_{IDMS} (i.e., the MU to which the RTP timestamp carried in the IDMS Settings packet refers[29]) at $p'^{k}_{IDMS,i}$ instant, which possibly does not match with p^{k}_{IDMS} instant (target presentation time included in the IDMS Settings packet). In such a case, the asynchrony, for each *n-th* MU, between the evolution of the local playout point of

[28] The playout rate deviations and their effect on multimedia synchronization were explained in Section 2.5.

[29] We are also assuming here that MUs are captured/generated periodically, e.g. with a constant transmission MU rate of $\theta \approx 25$ MU/s. Therefore, we can identify which MU the RTP generation timestamp included in the RTCP IDMS Settings packet (t'_{IDMS}) refers to, since the rate of advancements of RTP timestamps will be inferred by the PT field of the RTP media stream, and the RTP and NTP Timestamps can be mapped according to the info included in each RTCP SR (RFC 3550).

the *i-th* Sync Client ($p'^{k}_{IDMS,i}$) and the IDMS target playout point (p^{k}_{IDMS}) in each *k-th* group is given by $\Delta^{k}_{n,i}$:

$$\Delta^{k}_{n,i} = p^{k}_{IDMS} - p'^{k}_{IDMS,i} = p^{k}_{IDMS} - \left[p^{k}_{n,i} + \frac{1}{\mu^{k}_{n,i}} (MU^{k}_{IDMS} - MU^{k}_{n,i}) \right] \qquad \textbf{Eq. 8.5}$$

This asynchrony can be easily calculated as the time difference between the playout delay of the selected master reference for IDMS ($d^{k}_{master}=d^{k}_{IDMS}=p^{k}_{IDMS}-t'^{k}_{IDMS}$) and the current local playout delay for the *n-th* MU in *i-th* Sync Client belonging to the *k-th* group ($d^{k}_{n,i}$):

$$\Delta^{k}_{n,i} = (d^{k}_{IDMS} - d^{k}_{n,i}) \qquad \textbf{Eq. 8.6}$$

When DCS or M/S Scheme are used, the target playout point for IDMS will not be received in a RTCP IDMS Settings packet. Instead, it will be locally computed by each Sync Client in DCS, or by slave Sync Clients in M/S Scheme, independently of the other Sync Clients, using Equation 8.6.

Independently of the control scheme in use, the Sync Clients must adjust their playout processes to achieve IDMS. It can be done by following two possible reactive techniques. The first one is based on simple reactive actions such *'skips & pauses'* (aggressive playout adjustments), while the second one makes use of AMP (smooth playout adjustments). These adjustment techniques for IDMS are explained in Section 8.11.

8.10 Fault Tolerance

If a specific IDMS report (just one) sent by a Sync Client belonging to a specific group is lost, the IDMS control algorithm will not be drastically affected in any of the IDMS schemes. This is because the Sync Manager in SMS, all the distributed Sync Clients in that group in DCS, or all the slave Sync Clients in that group in M/S Scheme, will wait for the reception of the next IDMS report from that Sync Client, since the RTCP messages are sent regularly, as specified in RFC 3550.

If several successive IDMS reports from a Sync Client are lost, the Sync Manager, in SMS, or the distributed Sync Clients, in DCS, could have to wait for an excessive period in order to collect the overall IDMS timing. So, a control timer has been included to manage the triggering of necessary playout adjustments. Accordingly, the playout adjustments can be triggered either as a result of the detection of an asynchrony situation (exceeding an allowable threshold) or as a timeout event of this monitoring timer. In case of such a timeout event, the required IDMS adjustments will be calculated according to the collected reports from the

other Sync Clients. Every time IDMS setting instructions are sent to the Sync Clients (SMS) or directly performed by the Sync Clients (DCS), the timer is being reset.

The situation with loss of several successive IDMS reports is more problematic when using M/S Scheme. This is because in M/S Scheme only the master Sync Client reports on IDMS timing. Therefore, when the control timer runs out, the slave Sync Clients can decide to simply enforce IDMS adjustments according to the reference timing carried in the last received IDMS report from the master Sync Client.

As a summary, Figure 8.10 shows the flow chart of the overall IDMS algorithm, which is implemented in the Sync Manager for SMS, in all the Sync Clients for DCS, and in slave Sync Clients in M/S Scheme.

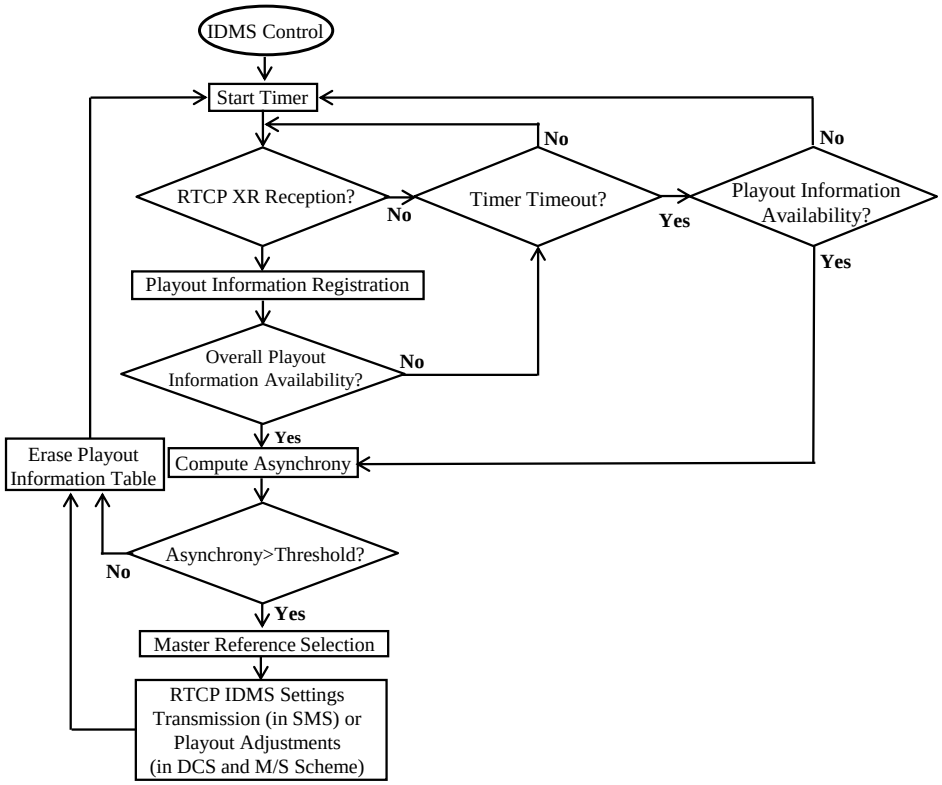


Figure 8.10. Flow Chart of the IDMS Algorithm.

8.11 Playout Adjustment Techniques

8.11.1 Aggressive Adjustments: Playout Skips & Pauses

If $\Delta^{k}_{n,i}>0$ (see Equation 8.6), the playout process of the *i-th* Sync Client belonging to the *k-th* group is advanced with respect to the selected IDMS reference. So, using aggressive adjustments, it must *'pause'* (i.e., stop playing) its playout process during $\Delta^{k}_{n,i}$ seconds to achieve IDMS, probably causing a *freezing effect* (see Figure 8.11). Consequently, the playout delay for the next MU, $d^{k}_{n+1,i}$, will be increased (i.e., $p^{k}_{n+1,i}$ will be delayed). Otherwise, if $\Delta^{k}_{n,i}<0$, the playout process of that Sync Client is lagged with respect to the IDMS reference. In that case, it must *'skip'* (i.e., jump or move forward) a certain number of MUs until the detected asynchrony is reduced to a lower value than the service/presentation time for one MU[30] (see Figure 8.11), thus probably causing a noticeable playout disruption and loss of information (as some video frames or audio samples will not be played out). This way, d_{n+1} will be reduced (i.e., p_{n+1} will be advanced).

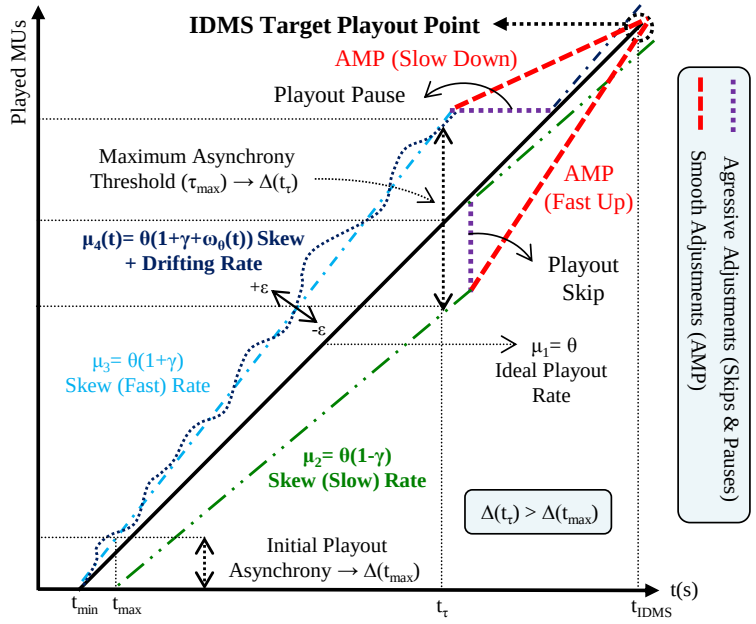


Figure 8.11. Playout Rate Imperfections & Playout Adjustments for IDMS.

[30] We are assuming that only entire MUs can be skipped, each one with a fixed presentation time of 1/θ ms/MU. Accordingly, the presentation time for MUs cannot be adjusted using our developed aggressive adjustment policy.

As shown in Chapter 11, those reactive playout actions (skips/pauses) will result in an overall synchronization status (within acceptable limits).

8.11.2 Smooth Adjustments: Adaptive Media Playout (AMP)

The above reactive playout adjustments could originate a noticeable degradation of the user perceived QoE. On the one hand, some relevant information may not be presented to the users, e.g. some important video scenes may not be visualized (due to the *skipped* MUs). On the other hand, a sensation of loss of continuity may be noticed, e.g. a *freezing effect* on the display (due to the *paused* MUs).

Playout disruptions can also be originated due to network and end-systems fluctuations (e.g., congestion). To mitigate the above effects, several AMP techniques have been proposed in the past (e.g., [Ish03b], [Kal04], [Chu07], and [Su09]). They consist of adjusting the media playout rate (i.e., playing the media faster/slower than normal), within perceptually tolerable ranges, to recover from undesired situations (e.g., buffer underflow/overflow or asynchrony situations) while providing glitch-free audio/visual quality. Previous works on AMP solutions have been mostly focused on improving the intra-media in audio and video streaming applications (e.g., [Chu07], [Su09]) and, occasionally, the inter-media synchronization quality (e.g., [Ish03b]). Also, playout speed modification has a precedent in traditional media broadcasting (although for a different purpose): motion pictures shot at a frame rate of 24 fps are shown on European PAL/SECAM (Phase Alternation Line / Séquentiel Couleur Avec Mémoire) broadcast television at 25 fps. Therefore, video frames are displayed during 1/25 s instead of 1/24 s, which corresponds to MUs dilation (speed up) of 4.2%, and it is typically done without audio time scale modification.

In this PhD thesis, a novel AMP technique is proposed to be used for IDMS purposes. The goal is to smoothly adjust the playout processes of the distributed Sync Clients every time an asynchrony threshold between their playout states is crossed, while minimizing long-term playout discontinuities.

The operation of AMP for video is more straightforward than for audio, since it simply consists of adjusting the display duration for each video frame. Nevertheless, AMP for audio involves signal processing in conjunction with time scaling techniques to stretch or widen an audio sequence, while preserving the pitch of the signal. In this PhD thesis we are only considering a single video stream, so we are not dealing with AMP for audio streaming.

The proposed AMP technique for IDMS is valid for VBR traffic patterns (but constant MU rate), as it is based on timestamps. Moreover, it can be applied in each one of the deployed IDMS control schemes. Its operation is as follows. Initially, the playout controller of each *i-th* Sync Client belonging to a specific *k-th* group must play out the buffered MUs at a non-adaptive playout rate given by $\mu^{k}_{n,i}=1/(s^{k}_{n,i})=1/(t'_{n+1}-t'_{n})$, being $s^{k}_{n,i}$ the service time of the *n-th* MU, as they were

generated by the Media Server. Accordingly, the generation timestamps of the incoming RTP packets (e.g., t'_{n+1} for the *n-th* RTP packet) will determine the normal playout rate in each Sync Client. Once each *n-th* MU finishes its presentation period, the next *(n+1)-st* MU must be played out, and the buffer occupancy must be updated.

As discussed, active Sync Clients (all of them in SMS and in DCS and the master in the M/S Scheme) include their current local playout point (t'^k_i, r^k_i, p^k_i) in each IDMS report they send to allow the Sync Manager (in SMS) or the distributed Sync Clients (DCS and M/S Scheme) to collect the overall playout status.

Using SMS, once an RTCP IDMS Settings packet is received, the target playout point for IDMS (d^k_{IDMS}, MU^k_{IDMS}) is registered and processed. At this point, the AMP process will attempt to either increase (speed up) or decrease (slow down) the video playout rate in order to minimize the detected asynchrony with the master, $\Delta^k_{n,i}$ (see Equation 8.6) among all the remaining MUs to reach the IDMS target playout point. This can be accomplished by means of increasing/decreasing the presentation period of all remaining MUs a value of $\delta^k_{n,i}=(\Delta^k_{n,i})/(MU^k_{IDMS}-MU^k_{n,i})$ seconds. This way, the local playout delay of the Sync Client can smoothly match the one of the IDMS reference in that *k-th* group (d^k_{IDMS}), as can be seen in Figure 8.11.

A key issue when performing AMP is to determine the allowed ratio within which the video playout speed can be varied without being annoying to the users' perception. Previous subjective studies have shown that playout speed variations up to 25% are often *unnoticeable* to users and, depending on the content and the frequency of the adjustments, variations up to 50% are sometimes *acceptable* ([Chu07], [Su09]). Subsequently, we assume in our tests that video playout adjustments up to 25% lead to unnoticeable quality impairments, and define a *playout factor* ($\varphi^k_{n,i}$) for each *n-th* MU in each *i-th* Sync Client belonging to the *k-th* group to specify this variation ratio. The value of this parameter is computed to get playout adjustments as smooth as possible, combining Equations 8.5 and 8.7, and using Equation 8.8:

$$p^k_{IDMS} = p^k_i + \frac{1}{\mu^k_{n,i}(1+\varphi^k_{n,i})}(MU^k_{IDMS} - MU^k_{n,i}) \qquad \textbf{Eq. 8.7}$$

$$\varphi^k_{n,i} = \frac{1}{1+(\delta^k_{n,i}\cdot\mu^k_{n,i})} - 1 \qquad \textbf{Eq. 8.8}$$

Note that if the calculated $\varphi^k_{n,i}$ is higher than 25%, it will be bounded to that maximum scaling ratio (i.e., $|\varphi_{max}| \leq 0.25$), so as not to degrade the users' perception. In such cases, the Sync Client could not achieve a fine synchronization. It may occur when the allowed asynchrony threshold (τ_{max}) is set too high or when there are not enough buffered MUs to smoothly distribute the detected asynchrony between them. To avoid such a situation, proper values for the initial playout instant (p_{ini}), buffering

delay, τ_{max}, the master selection policy (d^k_{IDMS}), and the IDMS target playout point (MU^k_{IDMS}, t'^k_{IDMS}) must be set.

The flow chart of the AMP algorithm using SMS is sketched in Figure 8.12[31]. Note that in this figure, the playout buffer model at the client side, with a capacity of *C MUs* (not in bytes), is simplified by grouping the functionality of the jitter (in which RTP packets are queued to compensate the effect of the network jitter and to de-packetize the encoded video frames), decoder (in which the encoded frames will temporarily wait for their decoding processes) and render or display buffers.

Unlike in SMS, in which the Sync Clients receive the necessary playout adjustments in RTCP IDMS Settings packets from the Sync Manager, in DCS and M/S schemes, the Sync Clients must locally compute (apart from carrying out) the required playout adjustments based on the received IDMS reports. Therefore, the AMP flow chart for DCS and M/S Scheme is very similar to the one in Figure 8.12 but, in these cases, the IDMS target playout point will be directly and locally computed by the Sync Clients, independently of the other Sync Clients in the same group, and not received in an IDMS Settings packet. Accordingly, the Sync Clients must choose the degree of the playout adjustments to achieve IDMS. On the one hand, high values of the playout factor (near the maximum limit, i.e. $|\varphi_{max}| \approx 0.25$) will result in a rapid synchronization status (depending, of course, on the allowed τ_{max}). On the other hand, low values of the playout factor will originate smoother adjustments (helping to avoid noticeability), but the overall synchronization status will be reached later.

In this work, a linear adjustment policy has been adopted (for all the considered IDMS schemes). The number of MUs involved in the AMP process (N^{AMP}) in DCS and M/S Scheme must be enough to allow an extremely deviated (advanced or lagged) Sync Client, with an asynchrony with the selected IDMS reference (d^k_{IDMS}) near the allowed threshold (i.e., $\Delta^k_{max} \approx \tau_{max}$), to adjust its playout timing without exceeding the maximum playout factor (i.e., $|\varphi_{max}| \leq 0.25$). That number is given by the following expressions:

$$N^{AMP}_{advanced} \geq \left\lceil \frac{\tau_{\max}}{\frac{1}{\mu^k_{n,i} \cdot (1-\varphi_{\max})} - \frac{1}{\mu^k_{n,i}}} \right\rceil ; \quad N^{AMP}_{lagged} \geq \left\lceil \frac{\tau_{\max}}{\frac{1}{\mu^k_{n,i}} - \frac{1}{\mu^k_{n,i} \cdot (1+\varphi_{\max})}} \right\rceil \qquad \textbf{Eq. 8.9}$$

[31] The dependency of the *k-th* group has been omitted in the notation of this figure. This is because the playout controller does not need to know about the group membership. It is the responsibility of the RTCP agent of each Sync Client to filter and send the RTCP messages from/to the group it belongs to.

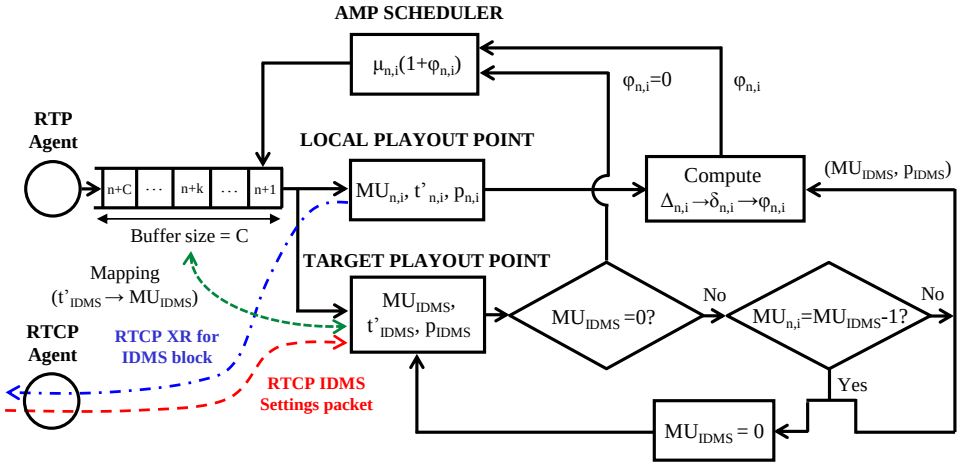


Figure 8.12. Operation of the AMP Technique for IDMS (in SMS).

In all the IDMS schemes, the AMP process will be finished once the IDMS target playout point (d^k_{IDMS}) is reached (i.e., once the playout delay of the Sync Client matches with the one of the master reference for IDMS) and will not be triggered again until a new out-of-sync situation is detected.

The smoothed and linear playout adjustments to acquire IDMS using the adopted AMP technique are illustrated in Figure 8.13. Note that when using DCS and M/S Scheme, slave Sync Clients will match the IDMS target playout point at different instants ($\Delta T \neq 0$ in Figure 8.13), because the asynchrony situation will not be simultaneously detected at each one of them (and they may further choose different IDMS target playout points). Using SMS, however, all the Sync Clients will be synchronized almost simultaneously with the IDMS target playout point (see Figure 8.11), because the required adjustments are indicated to them in RTCP IDMS Settings packets.

This way, using our AMP technique, the Sync Clients are able to achieve IDMS, using each one of the developed schemes, while avoiding long-term discontinuities in their playout processes, thus minimizing (or even avoiding any) users' annoyance.

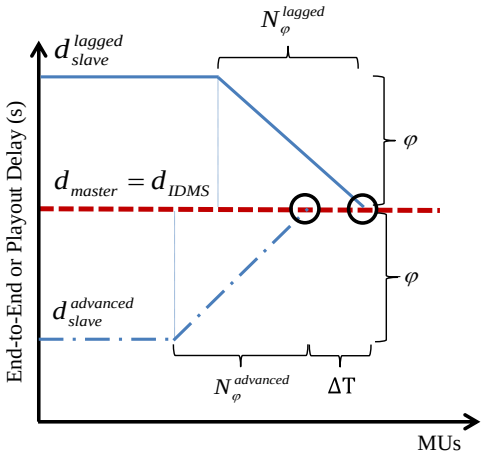


Figure 8.13. Smoothed and Linear Adjustments to Acquire IDMS when using DCS and M/S Scheme.

8.12 Coherence

This sub-section is only focused on the DCS-based operation of our IDMS solution and describes a novel technique for enabling better coherence (defined in Section 5.3) when this scheme is employed.

Let us assume that *"i-th Sync Client"* detects an asynchrony situation and starts its AMP process. During this period, the RTCP timer for that Sync Client expires and it sends an IDMS report including its (currently or recently adjusted) local playout point. It could be possible that *"j-th Sync Client"* is still waiting for that IDMS report from *"i-th Sync Client"* to complete the IDMS statistics and compute the asynchrony in the group they belong to. In such a case, the asynchrony situation will not be detected by *"j-th Sync Client"* because it has been (partially) corrected by *"i-th Sync Client"* prior to send its next IDMS report. This is not a serious constraint because, anyway, the overall asynchrony in that group will be kept below the allowed threshold. Nevertheless, if *"j-th Sync Client"* was not selected as the IDMS reference, its playout process would not be synchronized to the IDMS target playout point selected by the other Sync Clients in that group. This means that all the Sync Clients may not be simultaneously synchronized. Hence, there will remain a residual asynchrony among them, lowering the overall synchronization accuracy.

Consequently, a simple technique is proposed to solve this situation, thus providing better coherence when using DCS in our IDMS solution. It consists of using a bit of one of the fields reserved *"for future use"* in the IDMS report (see Figure 8.3) to indicate that an out-of-sync situation has been recently detected and corrected by the sender of this report (by setting it to *'1'*). This way, once that IDMS

report is received by the other *Sync Clients* belonging to the same group, they will be aware of such out-of-sync situation, and they will also adjust their playout processes to acquire a more fine-grained synchronization, using the most recent available IDMS timing information (e.g., the one computed the last time all the IDMS reports from that group were gathered).

The effect of this problem and the satisfactory responsiveness of the proposed technique are shown in Section 11.6.

8.13 Summary

In this Chapter, the rationale for using and extending RTP/RTCP for IDMS purposes has been provided. After that, the components of the designed IDMS solution in this PhD thesis have been presented. In particular, several of such components have been standardized (in RFC 7272), such as the proposed RTCP messages for IDMS, architectural solutions, and the SDP attribute for groups' management. Besides, the above components of our IDMS solution are fully compatible with other standard mechanisms (in RFC 7273) to inform about and negotiate the usage of related wall-clock sources for the involved sync entities in IDMS-enabled sessions. The standardization of our IDMS solution helps to ensure inter-operability and to promote the deployment in real scenarios.

Moreover, this Chapter has presented specific solutions for other required IDMS components, which have been left to vendor-specific implementations in the standard specifications for IDMS in both ETSI TISPAN and IETF. Examples are: control schemes for exchanging the information about IDMS, policies for selecting the IDMS reference to synchronize with, fault tolerance algorithms and playout adjustment techniques. Different alternatives for such components have been presented, and their feasibility and suitability for specific network conditions and application requirements have been discussed. This helps to efficiently deploy our IDMS solutions in a large variety of scenarios and use cases.

Finally, various implementation issues and operation aspects when performing IDMS have been also discussed.

By using the proposed RTCP messages for IDMS, highly accurate synchronization can be achieved, provided that precise mechanisms are available for: i) clock synchronization between the involved sync entities; ii) insertion and interpretation of the mapping between RTP and NTP-based timestamps in the involved RTCP messages; iii) computing delay differences between Sync Clients; and iv) performing the required playout adjustments.

Chapter 9

EARLY-EVENT DRIVEN (EED) RTCP FEEDBACK FOR IDMS

9.1 Introduction

In Chapter 5, it was concluded that SMS is, in general, the best control scheme for IDMS. However, it was also revealed that SMS performs worse than DCS and M/S Scheme in terms of interactivity, mainly due to two key issues. The first one is the required bidirectional communication processes between the Sync Clients and the Sync Manager to exchange the information about IDMS (see Figure 3.3.b). The second one is because the Sync Manager may have to adhere to bounded timing rules for sending IDMS setting instructions to the Sync Clients. This second issue is relevant in our RTP/RTCP-based IDMS solution when using Regular RTCP Feedback (RFC 3550). This is because Regular RTCP packets are exchanged in a pre-scheduled and inflexible manner, uniquely based on preserving the allowed traffic bounds specified in RFC 3550 (as explained in Section 7.5). There is no support for timely feedback that would allow to repair or to manage dynamic events of interest close to their occurrence. Accordingly, there may be a variable time lag (from few milliseconds up to several minutes in large-scale sessions) between detecting an event and being able to send an appropriate RTCP packet to handle it. Moreover, the RTCP packets may even not be received at the target side, since RTCP is sent over UDP, which does not provide a reliable channel.

Such constraints could limit the implementation of SMS in those IDMS use cases in which very stringent synchronization levels and timely responsiveness to dynamic events (e.g., detection of out-of-sync situations, rapid channel change delays…) are required.

A more efficient and strategic usage of the RTCP channel for IDMS is needed to overcome these issues. Subsequently, this Chapter presents further extensions to RTP/RTCP to enable higher flexibility, interactivity, dynamism and accuracy when

using such protocols for IDMS, while still adhering to the allowed RTCP traffic bounds specified in RFC 3550. In particular, novel Early Event-Driven (EED) RTCP reporting rules and feedback messages for IDMS, which in conjunction we call *EED RTCP Feedback for IDMS*, are specified with the goal of providing the following benefits: i) earlier correction of out-of-sync situations; ii) higher granularity for synchronizing the presentation of dynamically triggered media-related events (e.g., to ensure that important pieces of media content are simultaneously consumed by all the users); iii) ability of dynamically requesting IDMS setting instructions (e.g., in case of RTCP packet loss); iv) dynamic and rapid accommodation of latecomers in on-going sessions; and v) reduction of channel-change (i.e., zapping) delays. Such RTCP extensions for IDMS are based on the fact that not all the feedback reports are of equal importance and some of them need to be reported in a timely fashion.

The proposed EED RTCP Feedback for IDMS is applicable to and can have a potentially high impact on a wide spectrum of scenarios requiring IDMS, such as Social TV, networked multi-player games, synchronous e-learning, etc.

9.2 Immediate Initial RTCP IDMS Settings Packet

The same rationale for reducing the inter-stream synchronization delay in RFC 6051 (explained in Section 7.5) can be used for IDMS purposes. When using SMS in our RTP/RTCP-based solution, it would also be desirable the transmission of a nearly-immediate[32] RTCP IDMS Settings packet by the Sync Manager upon establishing a multimedia session.

If the Sync Manager is integrated within the Media Server, it must send the IDMS Settings packet just before or in parallel with the initial RTP data packets. If the Sync Manager is co-located within a Sync Client or a third party entity (that also needs to be an RTP receiver for that session), it must send the IDMS Settings packet as soon as it receives the initial RTP data packets from the Media Server. In either case, as the Sync Manager is a single centralized RTP entity, it is also allowed to transmit Early RTCP packets, as specified in RFC 6051.

This way, the Sync Clients can start consuming the media in a synchronized manner earlier, thus ensuring a reduction of the IDMS latency experienced by them.

[32] Note that in this work the terms (nearly-)immediate, close-to-instant and Early are used as synonymous. This is because the Sync Manager is a single centralized entity in the media session, and Early RTCP packets can be immediately sent by this entity without requiring a contention algorithm, as required for receivers in RFC 4585.

9.3 Dynamic EED Reporting of IDMS Settings

During the media session's lifetime, if Regular RTCP Feedback (RFC 3550) for IDMS is used, the Sync Manager may have to wait a nearly-complete RTCP reporting interval to be able to send a new compound RTCP packet (including an IDMS Settings packet) after detecting an event (e.g., an out-of-sync situation), which might potentially take several seconds (up to 5 s or even more), according to the timing rules in RFC 3550.

This issue is illustrated in Figure 9.1. In such a case, if an event (see blue circle in the figure) is detected just after the transmission of an RTCP packet (at instant $t_{r(1)}$), the next RTCP packet cannot be sent until the next randomized (over the scheduled transmission instant, $t_{d(2)}$) RTCP transmission time (at instant $t_{r(2)}$). The figure shows the worst case, in which the randomized RTCP report interval is near the upper limit (the possible reporting intervals are represented in squared red boxes), i.e.:

$$t_{r(2)} - t_{r(1)} \approx 1.5 \cdot [t_{d(2)} - t_{r(1)}] \qquad \textbf{Eq. 9.1}$$

Where:

- $t_{d(n)}$: *n-th* Scheduled (Deterministic) RTCP Transmission Time

- $t_{r(n)}$: *n-th* Real (Randomized) RTCP Transmission Time.

Therefore, the contribution of the Sync Manager delay (i.e., the time interval since an event is detected and an IDMS Settings packet to handle/repair it is sent) to the total IDMS latency (see Figure 9.1) becomes a serious barrier for those use cases requiring stringent synchronization levels (e.g., networked loudspeakers, or networked games).

Accordingly, the Sync Manager is also allowed to dynamically send Early RTCP IDMS Settings packets once detecting events throughout the duration of the session. This is illustrated in Figure 9.2. In such a case, an RTCP IDMS Settings packet is sent just after the detection of the event, despite that this moment is earlier than the next regular RTCP transmission time. Consequently, the IDMS latency is significantly reduced, mainly due to the fact that the Sync Manager delay has been minimized (due to the immediate transmission of the IDMS Settings packet).

Note that if *trr-int* (this attribute was introduced in Section 7.5) is set to zero, only one Early RTCP packet can be transmitted between two consecutive Regular RTCP packets in order to preserve the RTCP traffic bounds (RFC 3550). It means that an Early RTCP packet can only be sent if the previous transmitted RTCP packet was a Regular RTCP packet. Hence, after sending an Early RTCP packet, the RTCP reporting engine must schedule the sending time for the next RTCP packet by skipping the next Regular RTCP report interval (see dashed arrows in Figure 9.2), as in RFC 4585.

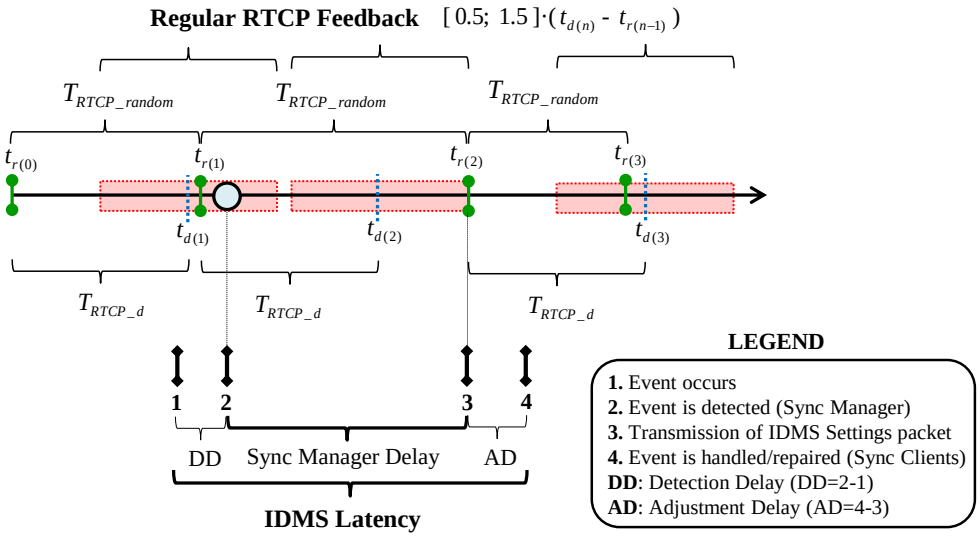


Figure 9.1. Use of Regular RTCP Feedback for IDMS.

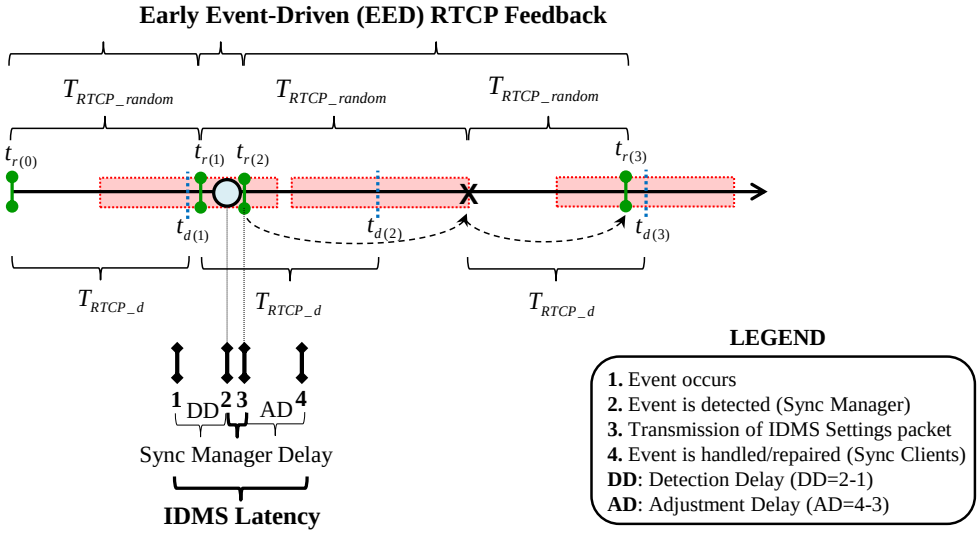


Figure 9.2. Use of EED RTCP Feedback for IDMS.

In case of a high frequency of events, setting an offset value for the RTCP report interval, by means of using the *trr-int* attribute, can help to save RTCP bandwidth (by restraining the transmission of too frequent Regular RTCP packets) while being able to use the (saved) bandwidth when events occur.

The dynamic EED reporting of IDMS Settings packets is also be very useful to provide playout hints for specific events that must be presented to all the involved users in a fine-grained synchronized way with the piece of content they refer to. Those events can be media-related events whose timing can be known in advance (e.g., commercials, start of the match in a sports event...), but the events' timing could be even unknown (e.g., a goal in a football match...) or dynamically triggered by either operators (e.g., a TV quiz show, in-game actions, interesting scenes…) or users (e.g., shared service control, interactive instant messaging…). Therefore, the use of EED RTCP Feedback for IDMS implies an interaction between the application-layer (through which operator or user generated events are triggered) and the transport/control layer (i.e., RTP/RTCP protocols) in order to translate the high-level (i.e., content-based or action-based) events into lower level calls (i.e., transmission of Early RTCP packets), as well as their alignment in terms of timelines. These are not severe issues, since the Sync Manager will be co-located with the Media Server in most implementations. However, this differs from the use of Regular RTCP Feedback for IDMS, in which the IDMS adjustments are purely based on packet-level timestamps.

We believe in the impact of the proposed EED RTCP Feedback for IDMS in real scenarios. For example, TV broadcasters are interested in including games, quiz shows (e.g., by offering prizes to the first user/s providing the right answer/s), or even bets, within the regular TV content. Concretely, some pilot tests were launched in the Netherlands, but these interactive TV services were rapidly cancelled. The main reason was the delay variability between delivery technologies and between destinations, which converted such services in a "game of chance", leading to unfairness between the home viewers (as the media content may be presented to them at different instants), and national legislations often prohibit to TV operators providing games of chance.

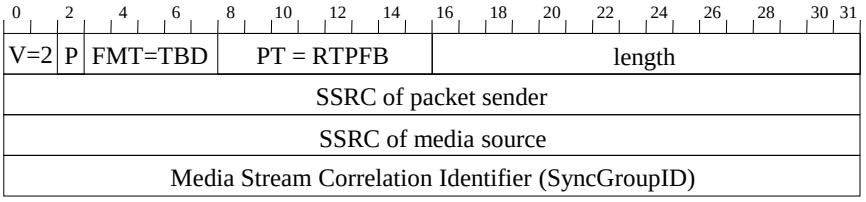
By enabling the EED RTCP Feedback for IDMS, the deployment of such services becomes feasible, as the targeted pieces of media content will be presented at all the viewers (almost) simultaneously. Therefore, such services can no longer be a game of chance, but rather a matter of users' skills and reaction times.

A similar mechanism exists in HbbTV. It consists of inserting *"do it now"* events as elementary streams into the MPEG-TS to allow the synchronization of dynamic events from extra applications (e.g., a question in an interactive quiz TV show, time-sensitive subtitles...) with the live DVB content. However, the proposed EED RTCP Feedback is not only valid to dynamically trigger local inter-stream synchronization (even though this is not tested in this PhD thesis), but also to enforce global and rapid IDMS adjustments in all the involved participants in a shared session, apart from being far more accurate.

9.4 Rapid (Re-)Synchronization Request

In an IDMS-enabled session, if the initial compound RTCP packet (including SR, SDES and IDMS Settings packets) is lost, the Sync Clients will not be able to synchronize the media playout until the report interval has passed, and the next RTCP packet can be sent. This is undesirable. RFC 6051 defines a new RTP/AVPF transport layer feedback message (this type of RTCP messages are defined in RFC 4585), called RTCP-SR-REQ, to request the generation of an Early RTCP SR, allowing rapid inter-stream (re-)synchronization.

A similar mechanism is proposed in this PhD thesis to be applied for IDMS purposes. A new RTP/AVPF transport layer feedback message, called RTCP-IDMS-REQ, is defined to request the rapid generation (and transmission) of an RTCP IDMS Settings packet from the Sync Manager (see Figure 9.3). The PT field of this RTCP message should be 205, as specified in RFC 4585, the Frame Message Type (FMT) should be assigned by IANA[33], and its length must be equal to 3. The SSRC of the packet sender field must indicate the Sync Client sending the packet, while the SSRC of the media source field must indicate the source of the media stream the Sync Client is unable to synchronize. In contrast to the RTCP-SR-REQ, in which the Feedback Control Information (FCI) part is kept empty, in the RTCP-IDMS-REQ it must carry the SyncGroupId (RFC 7272) of the group the sender of this message belongs to.



(TBD: To Be Determined)

Figure 9.3. RTCP IDMS-REQ Feedback Message.

Once a new RTCP-IDMS-REQ message is received by the Sync Manager, it must generate an Early RTCP IDMS Settings packet as soon as possible, following the Early RTCP feedback rules. This mechanism can also be employed if a Sync Client has not received IDMS Settings in a (configurable) long time interval.

The transmission of the RTCP-IDMS-REQ message may be repeated once per RTCP reporting interval if no RTCP IDMS Settings packet is received. Likewise,

[33] The proposed extensions for IDMS in this Chapter have been included in an IETF Internet Draft [Mon15], which will be presented at the next 93th (March 2015) IETF meeting.

the Sync Manager may ignore incoming RTCP-IDMS-REQ if its regular schedule for RTCP transmission will allow the Sync Clients to achieve synchronization within a reasonable time interval.

Although this mechanism is similar to the one for requesting rapid SRs in RFC 6051, it is especially necessary since, in most implementations, the IDMS Settings packets will not be regularly sent in each RTCP report interval, as RTCP SRs, but only when the detected asynchrony exceeds an allowable threshold.

9.5 Rapid Accommodation of Latecomers

In multi-party multimedia services, users may join and leave the session quite frequently. A user who joins a session in progress is usually called a latecomer. The support for and rapid accommodation of latecomers are key issues to enable dynamic IDMS-enabled sessions. This is another useful applicability of the proposed RTCP-IDMS-REQ message.

Once a latecomer joins an IDMS-enabled session, it must send an RTCP-IDMS-REQ message to the Sync Manager of that session. Then, the Sync Manager must send an Early RTCP IDMS Settings packet to the latecomer. Immediately after receiving the RTCP IDMS Settings packet, the latecomer will begin to play out the media stream in a time synchronized way with the other Sync Clients, thus becoming an additional member in the IDMS-enabled session, as shown in Figure 9.4. This will prevent from both long annoying startup delays and initial playout inconsistencies.

The timing diagram for the RTCP exchange processes is illustrated in Figure 9.4. It can be seen that, when using EED RTCP Feedback, the IDMS latency for latecomers (i.e., the time interval between joining the session and acquiring IDMS) can be significantly reduced mainly due to the fact that the Sync Manager delay (Δt_2 in Figure 9.4) can be minimized.

Two additional mechanisms could contribute to further reduce the IDMS latency (see Figure 9.4). The first one consists of employing priority mechanisms for the transport of RTCP messages, e.g. by adopting a Differentiated Services (DiffServ) policy, as in [Bgc10]. This would help decreasing the RTT delays and the loss probability for RTCP packets (out of the scope of this PhD thesis). The second one is based on the transmission of Early RTCP-IDMS-REQ messages by latecomers upon joining the session. According to RFC 6051, the delay since joining and sending an RTCP-IDMS-REQ message (Δt_1 in Figure 9.4) should not be reduced to avoid flooding of requests at specific time instants (e.g., at the time a broadcasted sport event begins).

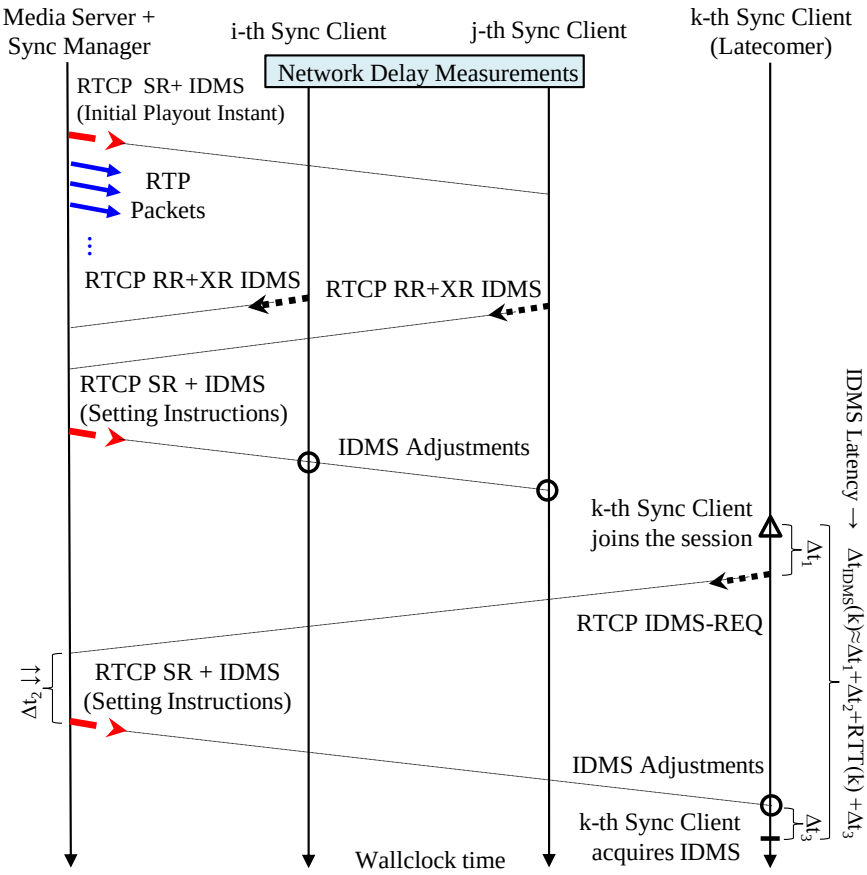


Figure 9.4. RTCP Message Exchanges for IDMS using SMS.

While in this PhD thesis we adhere to this standard compliant rule, an interesting future work is to investigate if this flash crowd effect is a real limiting issue in different large-scale SSM scenarios (e.g., networked quiz shows, gaming, IPTV...). Our initial assumption is that the upstream bandwidth availability by the Sync Clients (which is not used for other purposes) and the aggregation and re-distribution mechanisms by Feedback Targets (defined in RFC 5760) do not entail a real constraint for allowing the transmission of Early RTCP-IDMS-REQ messages by the Sync Clients. Moreover, it is assumed in RFC 6051 that all Sync Clients switch channels simultaneously, but even though using automated procedures (e.g., through notifications via the EPG in IPTV), this would not be a matter of a few seconds, but more probably of minutes.

9.6 Reduction of Channel Change Delays

Similarly, the transmission of Early RTCP-IDMS-REQ messages is also applicable for reducing channel change (i.e., zapping) delays in IDMS-enabled sessions.

Previous studies (e.g., [Ram13] and [Man13]) have shown that large channel change delays have a serious impact on the perceived QoE, thus being an obstacle for the wide adoption of IPTV services. Therefore, reducing channel changes delays is currently a hot research topic, with significant commercial relevance. In [Fuc08], [Ram11], [Ram13] and [Man13], several sources of channel change delays are identified, such as: multicast (join/leave) procedures, network delays and jitter, buffering techniques, media encoding settings, packet loss handling, and acquisition of the necessary reference information (e.g., Program Specific Information - PSI -, I-frames, encryption keys...) from the stream to start its consumption. Therefore, the optimization of such components, in conjunction with the provisioning of additional techniques, such as predictive tuning methods (by pre-joining channels), secondary tune-in streams, as well as allocation and assignment of auxiliary servers, is essential to contribute to the decrease of the channel change delays. Up to date, several solutions (e.g., [Fuc08], [Beg09], [Bgc10] and [Ram11]), even IETF standards (e.g., RFC 6285 [Ste11] and RFC 6659 [Beg12]), have devised specific solutions to help decreasing channel change delays when using RTP/RTCP protocols.

Moreover, as discussed in [Fuc08] and in RFC 6051 [Per10b], another key source of channel change delay is the required time for receiving RTCP packets, which are necessary to perform inter-stream synchronization. This is because media will not be played out until the involved streams can be synchronized, and this synchronization process must not contribute to further increase the channel change delays. Therefore, in conjunction with the above techniques, the RTCP timing rules from RFC 6051 should be employed to enable rapid (inter-stream) synchronization. As an example, the works in [Beg09] and in [Bgc10] made use of a rapid acquisition technique (by employing an auxiliary retransmission server), combined with Early RTCP Feedback reporting rules, to decrease channel change delays when joining on-going RTP multicast sessions.

The above discussion also applies to IDMS. Specifically, the relevance of channel change delays and their variability in IDMS-sensitive services is threefold. First, as for inter-stream synchronization, the required time to receive the IDMS setting instructions must not contribute to further increase the channel change delays. Second, apart from the magnitudes of channel change delays, their variability (i.e., the delay variation for each involved user) will also impact the IDMS performance. Third, when a group of users are watching IPTV together and they (simultaneously) change (or must change) to another channel, any playout time differences among them will also influence the resulting delay.

As a proof of relevance, the results of real-life measurements to determine the magnitude of channel change delays and of their variability are presented. Such

measurements were performed at TNO (Delft, the Netherlands) for three (anonymized) Digital TV (DTV) providers: two IPTV providers, one of them using VDSL2 (Very-High-Bit-Rate Digital Subscriber Line 2) access and the other offering a FTTH (Fiber To The Home) based service, and a third DVB-C (DVB Cable) provider (for the sake of comparison)[34].

Figure 9.5 shows the results of such measurements for each DTV provider setup when changing to both SD and HD channels. 20 repetitions were performed for each test, from which the T-distribution was used to calculate 95% confidence intervals. The accuracy of such measurements is around 20 ms, since a video camera of 50 fps was used to record and measure such delays (i.e., the time interval between pressing the button on the remote control and the media content being played out at normal speed).

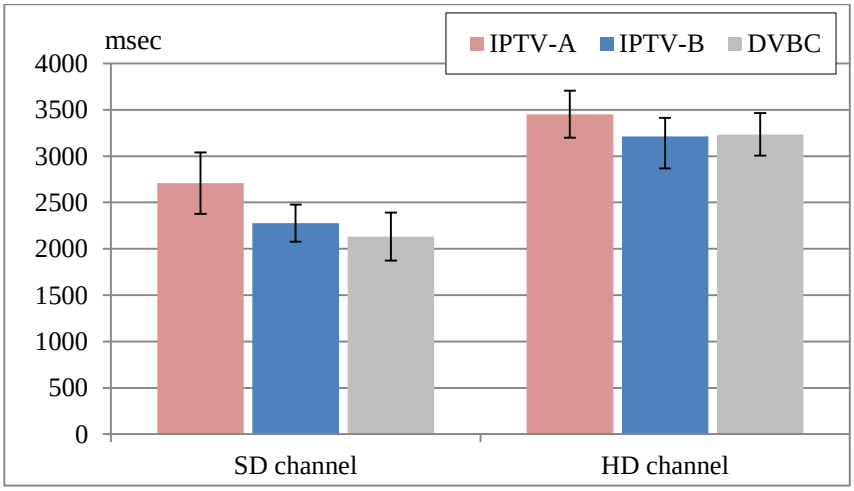


Figure 9.5. Average channel change delay for an SD channel and an HD channel, measured at three different DTV providers.

First of all, it can be seen that channel change delays were above 2 s in all the tested conditions. Such magnitudes of delays are easily noticeable, and probable annoying, for most users. The magnitudes of these values are in line with the ones reported in [Ram11] and [Man13] for commercial IPTV systems. However, what is really relevant for IDMS is the channel change delay variability when: i) the same setup (i.e., same quality channel and provider) is performed multiple times (as the

[34] We thank Harrie van der Vlag and Hans Stokking, researchers from TNO (Deflt, the Netherlands), for providing us the results of such measurements.

channel change delay varies per each user and per test); ii) using different quality channels provided by the same DTV operator (as the involved users in an IDMS-enabled session could be watching either SD or HD channels); and iii) using the same quality channel provided by different operators (as the involved users in an IDMS-enabled session could be subscribed to different DTV operators).

For example, the average delay for changing to an SD channel in the setup for IPTV-A operator was 2.7 s, with a 95% confidence interval between 2.4 and 3.05 s. This indicates at least a 95% chance that a new repetition of the same channel change test will result in a delay between these values. The delay variation between DTV providers was in the order of 0.5 s. This may be mostly due to the different hardware and software components in each of them. Moreover, it can be seen that changing to an HD channel took about a second longer than changing to an SD channel in each of the considered DTV operators.

Such delay variability ranges have a significant negative impact on IDMS. Therefore, in this context, the use of EED RTCP Feedback for IDMS is very beneficial because: i) it significantly reduces the time needed to receive the initial RTCP IDMS Settings packet; and ii) it enables the compensation of the delay differences when changing channels.

We also considered the transmission of just-in-time RTCP IDMS Settings in parallel with each I-frame as an alternative solution to help reducing channel change delays in IDMS-enabled sessions. This would allow to concurrently align both the necessary synchronization and reference information ([Fuc08], [Ram11]) to start playing out the stream. However, this approach presents several drawbacks. First, it requires a direct control over the application layer to identify the transmission of each I-frame. Second, it implies a continuous modification of the RTCP reporting times. This is because the frequency of I-frames can significantly differ from the one of the RTCP report interval. So, it may imply breaking the RTCP timing rules (probably exceeding the allowed traffic bounds). Third, the frequent reception of IDMS Settings packets by the Sync Clients can increase their computational load and lead to too frequent, unnecessary (because their playout timing is still relatively in-sync), and probably annoying, playout adjustments. Moreover, it cannot be obviated that compound RTCP packets include other relevant information apart from IDMS statistics, so this method could have a negative impact on the reporting of this extra information. Accordingly, the use of explicit RTCP-IDMS-REQ messages is a more appropriate approach.

Similarly, as discussed in Section 7.5, the inclusion of in-band synchronization metadata with RTP header extensions is allowed in RFC 6051 to enable rapid inter-stream synchronization. For that purpose, originating NTP-based timestamps can be inserted into the headers of each RTP packet (or, at least, into the header of the first packet) containing a key frame or RAP. Such information must be also included for each of the involved streams to be synchronized, probably carrying out other media

types (e.g., audio), to allow their temporal alignment. In [Fuc08], it is claimed that this approach can also contribute to decrease channel change delays. However, the use of a similar method is not optimal for IDMS, because the reference information for performing IDMS is based on reception and presentation timestamps reported by Sync Clients, which can further vary for each of the involved groups of Sync Clients, unlike the reference information for performing inter-stream synchronization, which is based on originating timestamps (unique for each media stream and for all the Sync Clients).

9.7 Summary

In this Chapter, our IDMS solution has been extended by devising a more strategic and efficient usage of the RTCP channel for IDMS. In particular, novel EED RTCP Feedback reporting mechanisms have been presented to enhance the performance of our IDMS solution in terms of interactivity, flexibility, dynamism and accuracy, while still adhering to the allowed RTCP traffic bounds specified in RFC 3550. The different Sections of this Chapter have described the ability of the EED RTCP Feedback to rapidly react on dynamic situations, such as correction of out-of-sync situations, loss of RTCP packets, accommodation of latecomers or channel change delays, as well as to enable finer granularity for synchronizing media-related events.

The proposed EED RTCP Feedback for IDMS is applicable to and can have a potentially high impact on a wide spectrum of scenarios with commercial relevance, such as Social TV, networked multi-player games and synchronous e-learning.

Finally, the proposed extensions for IDMS in this Chapter have been included in an IETF Internet draft [Mon15], which is going to be presented at the next 93th (March 2015) IETF meeting.

Chapter 10

EVALUATION METHODOLOGY AND PROTOTYPE IMPLEMENTATION

10.1 Introduction

This Chapter presents the evaluation methodology that has been followed and the prototypes that have been implemented in this PhD thesis. First, the rationale on using both a simulation and a real media framework for evaluating our IDMS solution is provided in Section 10.2. After that, the prototype implementation in a simulation framework, concretely in Network Simulator 2 (NS-2) [35], is described in Section 10.3. Finally, the prototype implementation in a real media framework, concretely in GStreamer[36], is described in Section 10.4.

10.2 Evaluation Methodology

According to the schedule of the design process, the prototype implementation and evaluation processes have not been performed at a single stage, but repeated for each individual component of the IDMS solution under design (described in Chapters 8 and 9), as well as for the global IDMS solution at a later stage. The workflow of the PhD thesis, emphasizing the evaluation phase, is sketched in Figure 10.1.

In order to validate the performance of the IDMS solution, two prototypes have been implemented, the first one in a simulation framework and the second one in a real media framework.

[35] Network Simulator (NS-2): http://nsnam.isi.edu/nsnam/.
[36] GStreamer: http://gstreamer.freedesktop.org/.

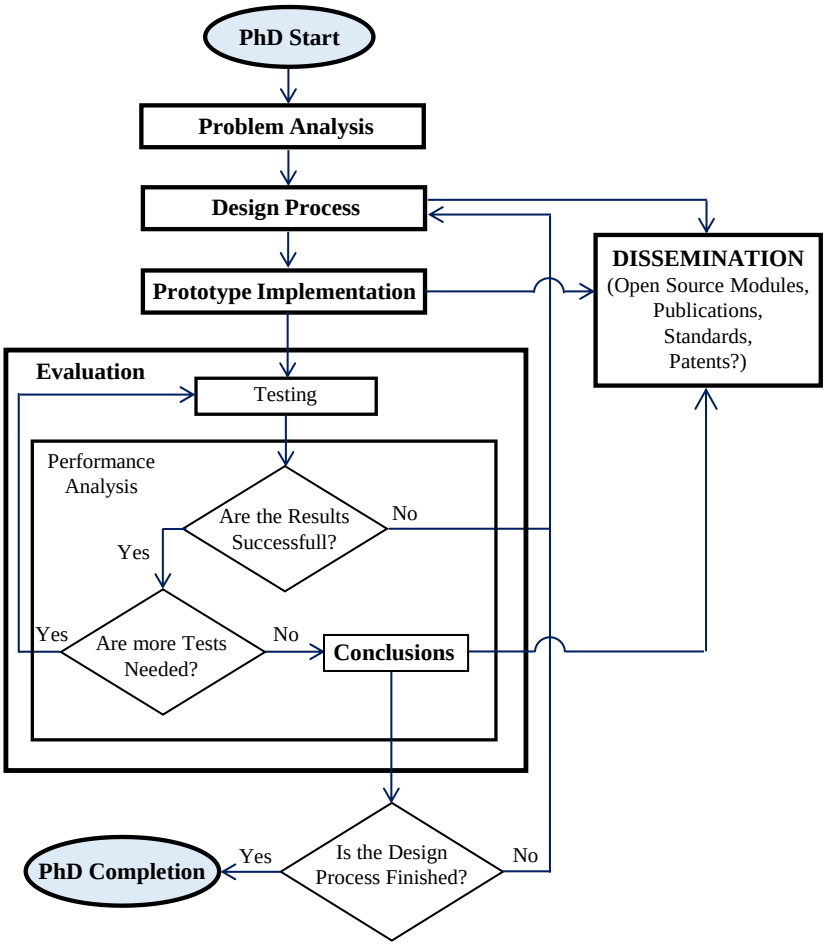


Figure 10.1. Workflow of the PhD thesis: Evaluation Phase.

By using network simulation techniques, researchers can assess the feasibility and suitability of their proposals, as well as to compare them with another existing ones, in heterogeneous networked environments, without the need for any physical scenario. The use of a simulation framework allows testing various proposals that might be very difficult, or even impossible, to evaluate in real systems, mainly due to: i) the unavailability of the necessary equipment and infrastructure; ii) the temporal and spatial requirements; and iii) the necessary involvement of many users (especially when testing application-layer solutions). By using network simulation techniques, the repeatability of the experimentation becomes much more feasible than using real-world assessments. Likewise, in each of the simulations, or even during simulation time, various conditions or parameters of the system in which the proposal is evaluated (e.g., network topologies, links' capacity, network load,

encoding parameters, protocols, number of nodes...) can be modified in a controlled manner to assess the behavior of such proposal in different cases. In addition, such forced situations and multiple tests do not imply an interruption of the normal operation (e.g., loss of service) or any prejudice (e.g., links or equipment failures) to the system under test, as it could do when using real systems. Moreover, the overall system can be monitored and controlled from a single computer. This enables higher modularity and flexibility for double checking assumptions and for identifying critical issues during the design and evaluation processes of the targeted proposals.

Within the context of this PhD thesis, network simulation techniques can facilitate the assessment of the performance and consistent behavior of each of the designed components (i.e., protocols, schemes, algorithms and adjustment techniques) of our IDMS solution, by providing measurements of networking-related aspects and objective QoS metrics (e.g., delay, jitter, packet loss, traffic overhead, buffer occupancy levels...), as well as of delay differences (asynchrony) between the involved Sync Clients. Therefore, the use of a simulation framework can provide valuable information about the right direction of the research work in this PhD thesis. From among the variety of existing network simulators, NS-2 was chosen as the best candidate for implementing a prototype of our IDMS solution, as explained in the next section.

However, even though considering the many advantages and convenience of simulation tests, they cannot definitively validate the performance of our IDMS solution. In order to confirm the effectiveness of the different components of our IDMS solution and the synchronization accuracy that can be achieved, real-world assessments are also required. That is why we also decided to implement a prototype in a real media framework. Moreover, the implementation in a real media framework allows not only to carry out an objective testing but, most importantly, a subjective testing. This is a key stage that will definitively determine the benefits provided by our IDMS solution, which cannot be obtained through simulation tests. Subjective evaluation tests can be targeted to determine the user satisfaction (QoE) when enabling or not our IDMS solution in shared media experiences, to analyze the effects on the QoE of different levels of out-of-sync situations and how they are avoided when enabling our IDMS solution. Moreover, subjective testing can assess the impact of IDMS on the feeling of *"networked togetherness"* or *"fairness"* between the involved users, among other many relevant aspects.

GStreamer was chosen as the most appropriate real media framework for developing our IDMS solution.

As a summary, the evaluation methodology employed in this PhD thesis and the metrics that can be assessed in each of the prototypes are sketched in Figure 10.2.

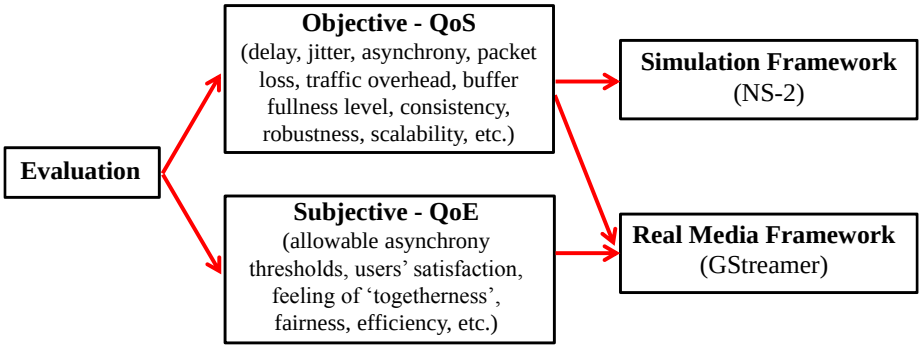


Figure 10.2. Evaluation Methodology in this PhD thesis.

10.3 Prototype Implementation in NS-2

This Section describes the prototype implementation in NS-2. First, an introduction to NS-2 and to the existing implementations of RTP/RTCP for that simulator is provided in Section 10.3.1. After that, the new RTP/RTCP module for NS-2, following strictly RFC 3550, developed in this PhD thesis, is presented in Section 10.3.2. In Section 10.3.3, the extensions to that NS-2 module to implement our IDMS solution are briefly introduced. Then, the developed playout buffering policy to guarantee intra-media synchronization in NS-2 is presented in Section 10.3.4. Finally, the integration of our NS-2 module with other multimedia applications to constitute a toolset for video streaming evaluation is described in Section 10.3.5.

10.3.1 Introduction to NS-2

Nowadays, NS-2 has become a widely adopted simulation tool by the networking community for evaluating various components of communication systems, such as network technologies, protocols, algorithms, etc. The core of the simulator is written in C++ language, while an object-oriented variant of Tcl (Tool Command Language) scripting language, called oTcL, is used for the configuration of the simulation scenarios.

Since its inception, NS-2 has been under constant improvement and, at present, it includes modules for the evaluation of heterogeneous network architectures, such as Mobile IP networks, WLANs (Wireless Local Area Networks), sensor networks, satellite networks, and many others. Additionally, it contains modules for numerous networking components such as MAC (Media Access Control) layer protocols, unicast and multicast routing algorithms, transport layer protocols, traffic source behavior, queue management mechanisms, statistics measurement, etc. Despite this variety, oftentimes researchers need to adapt the existing NS-2 modules to their requirements or incorporate new modules to evaluate further functionalities not

included in the built-in NS-2 code. The simulator is open source; hence, such modifications and extensions are possible. Therefore, the above reasons made us choosing NS-2 as the simulation tool for undertaking the research in this PhD thesis.

10.3.2 Native RTP/RTCP Implementation in NS-2

Generally, protocols are implemented in NS-2 as *Agents*. These agents represent end-points where packets are constructed or consumed, and they can be used for the implementation of protocols at various layers. In the original NS-2 code, RTP and RTCP protocols are implemented as the *RTP Agent* and the *RTCP Agent* classes, respectively. These two classes derive from the *Agent* class, and are implemented in the *rtp.cc* (located in *~ns/apps/rtp.cc*[37]) and *rtcp.cc* (*~ns/tcp/rtcp.cc*) files, respectively, as can be seen in Figure 10.3. The *RTP Agent* is responsible for transmission and reception of RTP packets, whereas the *RTCP Agent* is responsible for transmission and reception of RTCP packets. The *RTP Session* class (located in *~ns/common/session-rtp.cc* file) mainly includes the functionalities for RTCP feedback report building and for registering the relevant information from the participants in the RTP Session. It also defines the procedures for the session initialization, RTCP report interval calculation, RTP transmission rate, packet size setting, etc. During the evolution of the session, when data from a new RTP source is received, the *RTP Session* includes the parameters and statistics from this source in its sources' information table. Similarly, when RTCP reports from RTP receivers are received, the *RTP Session* includes their parameters and statistics in its receivers' information table. All the previous C++ files use *rtp.h* as the header file, which is located in *~ns/apps* directory. Such C++ files are shown in red boxes in Figure 10.3.

This native NS-2 module for RTP/RTCP protocols is quite generic. It does not include many of the attributes specified in RFC 3550 [Sch03], while some other implemented ones do not strictly meet the RFC 3550 specification: i) it does not define all the RTCP packet types; only RTCP SR packets are included, but its format is not complete (it does not include the PT field, the number of sent packets field, the number of sent octets field, etc.); ii) since RR packets are not defined, neither QoS metrics (e.g., jitter, delay, or loss rate) monitoring nor reporting are provided; iii) the same packet header is used to generate both RTP and RTCP packets; iv) the fields of this packet header are specified using incorrect variables' types and sizes; v) it does not work properly for multicast transmissions; vi) it does not support multiple multicast streams on the same node; vii) the RTP Agent is only capable of generating CBR traffic (i.e., VBR traffic patterns are not supported), etc.

[37] ~ns refers to the local ns2.XX directory in the source code of the simulator, where XX denotes the installed version.

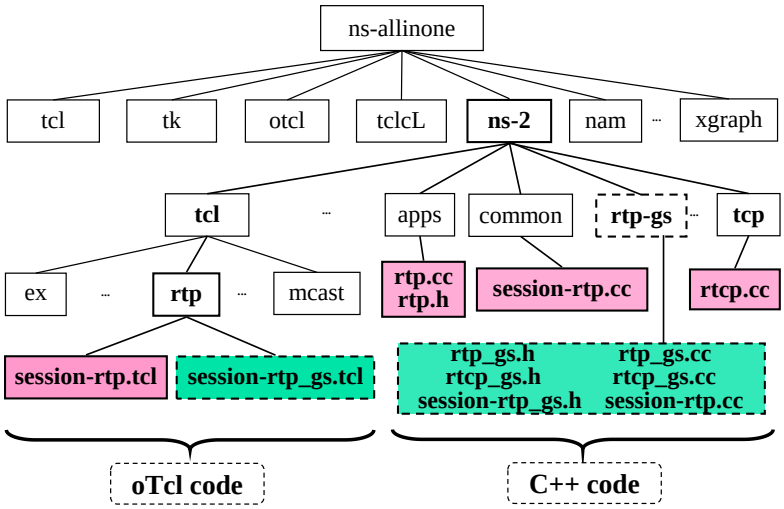


Figure 10.3. NS-2 Directory Structure.

Consequently, as simulations rely on the accuracy of the proposed models, we decided to develop a new NS-2 module with a more precise and complete implementation for RTP/RTCP protocols, by including all the features specified in RFC 3550 that were not originally considered or accurately implemented in the legacy NS-2 code.

10.3.3 Other Existing RTP/RTCP Implementations in NS-2

Apart from the native RTP/RTCP module for NS-2, two additional implementations of such protocols, including improvements to that native code and new functionalities for specific purposes ([Car09] and [Bou08]), have been found. On the one hand, in [Car09], new RTP and RTCP Agents were defined to provide loss and jitter control for video streaming applications with QoS support. In addition, new independent data structures were defined to generate the RTP and RTCP packets. These data structures contain more differentiated fields than the one in the native code, but their sizes are not correct and some fields of these data structures are not specified in RFC 3550. On the other hand, in [Bou08] new extensions to the legacy RTP/RTCP code in NS-2 were added in order to: i) provide additional features related to QoS measurements (concretely, loss and jitter control); and ii) employ TCP Friendly bandwidth share behavior for multicast streaming applications.

Acknowledging the new functionalities and enhancements of such implementations compared to the native code, various features were still missing. So, we decided to implement a new module for NS-2 with a full and accurate

implementation of RTP/RTCP protocols (following strictly the RFC 3550). Concretely, our new RTP/RTCP module includes the following functionalities: i) definition of all the types of RTCP packets with their exact format (SR, RR, SDES, BYE and APP packets); ii) network-level metrics (such as end-to-end delay, jitter, RTT, throughput and packet loss) monitoring, processing and registering in simulation time; iii) capability of processing any kind of application traffic pattern supported by the application-layer in NS-2; iv) support for multiple multicast streams on the same node; v) accurate implementation of the RTCP reporting rules; and vi) compatibility with the legacy code. The functionalities of the new developed RTP/RTCP code for NS-2 are briefly described in the next sub-section.

10.3.4 New Developed RTP/RTCP Module for NS-2

In contrast to the other two discussed implementations, our new RTP/RTCP module can be included together with the other built-in NS-2 modules, without needing to replace the legacy RTP/RTCP code. As the directory structure for the native RTP/RTCP implementation in NS-2 is a bit confusing and dispersed, we have collocated our source files in a more coherent way, as can be observed in Figure 10.3 (dotted line green boxes). The C++ code has been located in *~ns/rtp_gs* directory and the oTcl code has been located in *~ns/tcl/rtp_gs* directory (*gs* stands for *"group synchronization"*). Likewise, a new header file has been included for each C++ file (see Figure 10.3). Moreover, independent data structures have been included to generate the newly defined RTP and RTCP packet types, thus distinguishing them from the packets generated by the native code.

10.3.4.1 RTP Management

In the *rtp_gs* files, we have re-defined the native RTP packet header, with the exact format specified in RFC 3550. Moreover, we have re-implemented the *RTP Agent* and improved it in order to be capable of transmitting any kind of traffic pattern supported by the application layer (such as Pareto, CBR, Exponential or Traffic Trace Files generated from real multimedia applications) in contrast to the native RTP Agent, which is only capable of sending CBR traffic. Therefore, new RTP packets will be generated based on the reception of new data blocks from the application layer. If the data blocks are larger than the Maximum Segment Size (MSS), which can be configured in each simulation, the RTP Agent fragments the data payload into several RTP packets. Next, the packet headers, with the appropriate fields, are filled and the RTP packets are sent to the destination/s in a unicast or multicast way.

When incoming RTP packets arrive to the receiver *RTP Agent*, it passes them to the *RTP Session* instance in order to process, register and update the relevant statistics from that RTP Sender. Such statistics include: the total number of packets and octets that have been received, the cumulative number of lost packets, the

highest sequence number that has been received and the jitter value for that RTP packet. Such statistics are also registered in output trace files for post-processing.

10.3.4.2 RTCP Management

In *rtcp_gs* files, a new common header for the RTCP packets has been defined. This differs from the native code, which makes use of the same packet header to generate both RTP and RTCP packets. Likewise, new data structures for each RTCP packet have been defined, with the exact format specified in RFC 3550.

Accordingly, the *RTCP Agent* has also been re-implemented and extended to be able to manage the sending and receiving processes of the newly defined RTCP messages. Likewise, new data structures and output trace files have been added to register the statistics from the incoming RTCP packets in each simulation. In addition, control timers and appropriate functions have been defined in order to accurately implement the dynamic RTCP timing rules (explained in Section 7.5).

This NS-2 module with a full and accurate implementation of RTP/RTCP is publicly available at our website[38], and being currently used by various international researchers[39]. More specific details about such NS-2 implementation, and about the configuration of simple simulation scenarios, can be found in [Mon10a] and [Bor11a].

10.3.5 Integration of the IDMS functionality

The new developed RTP/RTCP module has been extended to include an optional functionality with all the components of our RTP/RTCP based IDMS solution: i) the RTCP extensions for IDMS; ii) the necessary elements to enable the adoption of each one of the control schemes; iii) timers for controlling various IDMS-related aspects (e.g., feedback reporting, fault tolerance algorithms, coherence technique…); iv) a proper playout buffering policy to provide intra-media synchronization solution and to enable the adjustment of the IDMS timing; v) the algorithms for monitoring, reporting and comparing the IDMS timing of the distributed Sync Clients; vi) the reactive techniques (playout adjustments actions) to achieve synchronization; etc.

Likewise, it is also important to emphasize that reporting on RTP presentation times is supported in our simulation-based prototype, because of the availability of a full control over the buffering and rendering processes at the client side. Finally,

[38] http://personales.gan.upv.es/~fboronat/.

[39] We have not included a download counter at our website, but our paper presenting this RTP/RTCP module for NS-2 has been downloaded more than 365 times, according to the ACM DL (Association for Computing Machinery Digital Library) website: http://dl.acm.org/citation.cfm?id=1808184. Moreover, many researchers have contacted us to thank and/or ask us for further instructions to install and configure our developed NS-2 module.

wall-clock synchronization between the involved sync entities can be guaranteed (with very high accuracy) using our simulation-based prototype, since all of them can make use of the same reference clock (provided either by an external clock source or by the global scheduler clock of the simulator) for inserting and interpreting timestamps.

10.3.6 Playout Buffering Policy in NS-2 (Intra-Media Synchronization)

Although NS-2 supports various (router) queueing policies, such as First-In First-Out (FIFO) with DropTail, Random Early Detection (RED) or priority-based policies, it does not include any playout buffering policy. Accordingly, the incoming packets are destroyed upon reaching the targeted end-point agents, after registering the packets' identifiers, their reception time and the jitter values for each of them in output trace files (if desired). Therefore, we decided to design our own playout buffering in NS-2 to able to: i) enable intra-media synchronization, by smoothing out the effect of network jitter; and ii) compensate the delay differences between the Sync Clients by adjusting their playout timing (as described in Section 8.11) when the IDMS functionality is enabled.

Typical intra-media synchronization solutions mainly consist of storing the incoming data packets in playout buffers before playing them out in the same temporal order they were generated. The goal is to obtain a sequence of playout instants, such that the original temporal relationships between successive MUs can be maintained (e.g., uniformly spaced, if CBR traffic is used). This phenomenon is sketched in Figure 10.4, which represents the transmission, reception and playout times in a bursty stream of MUs (the example is also valid for a continuous stream).

For simplicity, let us assume that MUs are conveyed into single packets. Let t_n, $r_{n,i}$ and $p_{n,i}$ be the time instants when the *n-th* MU (of a specific burst) is transmitted, received and played out by a specific *i-th* Sync Client, respectively. The *network delay* or latency of the *n-th* MU for that *i-th* Sync Client is given by the difference between its reception and transmission instants: $l_{n,i}=r_{n,i}-t_n$. The variability of the network delay can be seen in the middle row of Figure 10.4. The effect of network jitter can be alleviated by temporarily storing the incoming packets (conveying MUs) into the playout buffer, and then pushing them at the original transmission rate, even though at the expense of adding extra delay to the multimedia service. The *playout or end-to-end delay* of the *n-th* MU for the *i-th* Sync Client, $d_{n,i}$, is given by the time difference between its playout and its transmission instants: $d_{n,i}=p_{n,i}-t_n$. It should be uniformly kept for each couple of *n-th* and *k-th* MUs in a smooth playout process, i.e. $(p_{n,i}-p_{k,i})\approx(t_n-t_k)$, at least for a group of packets corresponding to the same burst. The playout delay is mainly given by the sum of network and buffering delays (if encoding, packetization, depacketization and decoding delays are considered as negligible), which instead are both variable. The playout controller of each *i-th* Sync Client will schedule the playout time of the successive MUs at $p_{n+1,i}=p_{n,i}+s_{n,i}$, where $s_{n,i}$ refers to the service time of the *n-th* MU, which is given by the difference

between the timestamps of the *n-th* and *(n+1)-st* MUs (if constant MU rate is assumed, and the source transmits θ MU/s, $s_{n,i}$ should be equal to $1/\theta$ seconds). The playout delay needs to be continuously adapted in order to maintain a trade-off between handling of late packets and the tolerable additional delay to the multimedia service. On the one hand, if $d_{n,i}$ is increased, by means of increasing the buffering delay, then less packets will be discarded due to late arrival, but more delay will be added to the multimedia service. On the other hand, a reduction in $d_{n,i}$ turns out in less delay but higher packet discarding rate. Over the last years, significant research efforts have been devoted to issues about buffering the lower amount of data as possible, or introducing lower playout delay, without affecting the QoS.

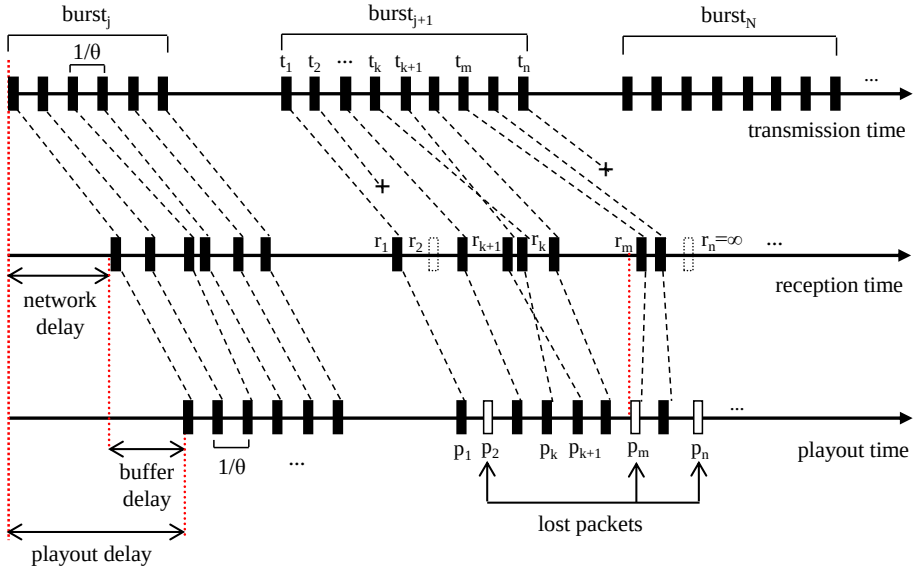


Figure 10.4. Intra-Media Synchronization (Playout Buffering Policy).

Without loss of generality, we assume that $r_{n,i}=\infty$ for each lost packet due to the network conditions (e.g., packets *2* and *n* of the middle burst in Figure 10.4). Furthermore, the playout buffering policy usually discards all packets that arrive later than their scheduled playout instants, i.e., packets with $l_{n,i}>d_{n,i}$, as for packet *m* in Figure 10.4. Up to date, various strategies have been devised to handle/conceal these losses [Lao02]. One alternative is to employ prediction methods based on the information from the previous and successive received MUs. Other possible options consist of repeating the playout of the previous MU (suitable for video streaming) or maintaining an empty period during the scheduled playout point for the lost MU (suitable for audio streaming). Less often, late packets are not discarded, but the playout of the previous MU is stretched until the arrival of the late ones.

In our simulation-based prototype, we have implemented playout buffers with a configurable capacity. When a packet is received, the playout buffer checks if the incoming MU can be accommodated or if it should be dropped. An incoming MU is not added to the buffer if its playout time has already elapsed. The playout buffer policy also supports the following dropping policies [Lao02]: i) when the buffer is full, the newly received MU will be discarded; and ii) when the buffer is full, the first MU to be dropped will be the next to be played out (i.e., the buffer would discard the oldest MUs, keeping the more recent ones). Finally, the buffering policy is capable of reordering MUs such that they are played out in the same order in which they were generated (using the sequence numbers and timestamps contained in each RTP packet's header), as can be seen for packets *k-th* and *(k+1)-th* in Figure 10.4.

Moreover, we have also include a functionality that allows setting different playout rate imperfections (i.e., skews and drifts) in each Sync Client. As discussed in Section 2.6, this can have a serious impact on the different types of multimedia synchronization, such as intra-media synchronization and IDMS.

As a summary, the graphs in Figure 10.5 represent the transmission time of MUs of a media stream (black solid line), their reception times in two different Sync Clients (green and red curves), and their playout times in such Sync Clients (blue and orange curves). It can be observed that, in Sync Client 2, the designed playout buffering policy (orange curve) handles the possible packet losses (*2-th* and *6-th* packets in the figure), by repeating the playout of the previous MU. Moreover, late packets (*m-th* packet in the figure) are discarded by the buffering policy, while out-of-order packets are re-ordered if their scheduled playout instant have not been elapsed yet (*4-th* and *5-th* packets in the figure). It can also be observed that if the initial buffering delay is fixed for all the Sync Clients (b_{ini} in the figure), there will exist an initial playout asynchrony between them (see $A(t_2)$ in the figure) due to the variability of the network delay between the Sync Clients and the Media Server. Moreover, if the Sync Clients present playout rate imperfections (in the figure the nearest/furthest Sync Client is the fastest/slowest one), this asynchrony will increase as the session advances in time (in the figure $A(t_4)>A(t_3)>A(t_2)$).

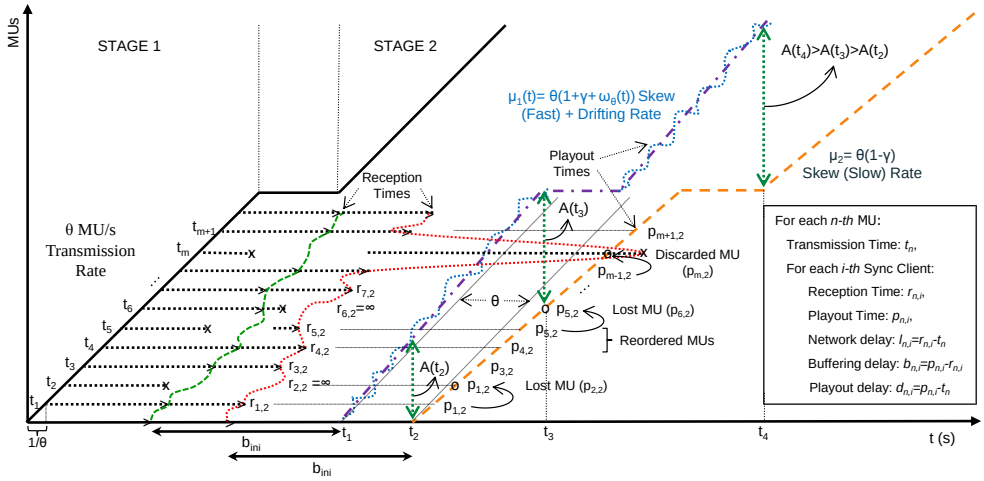


Figure 10.5. Playout Buffering Policy and Rate Imperfections.

10.3.7 Video Quality Evaluation Toolset

The developed RTP/RTCP module for NS-2 has been also combined with the necessary multimedia tools (such as video encoders/decoders, video players, trace files generators and video quality assessments programs) to constitute an advanced toolset for video quality evaluation. By using the capabilities of our RTP/RTCP module for NS-2, this toolset allows the measurement of network-level QoS metrics (such as throughput, delay, jitter or loss rate) in simulation time. Moreover, as it allows the transmission of real video files and their reconstruction and playout at the receiver side, the measurement of application-level objective video quality metrics (such as Peak Signal to Noise Ratio or PSNR, Structural Similarity or SSIM, and Video Quality Metric or VQM) and subjective metrics (Mean Opinion Score or MOS) is also supported.

The development of this toolset was motivated by the capabilities offered by Evalvid [Kla03], which is a framework for video quality evaluation over real and simulated scenarios. Evalvid supports the measurement of packet/frame jitter, loss rate, PSNR, as well as a static PSNR to MOS metrics mapping. Up to date, Evalvid framework has been integrated into several network simulators, such as NS-2 (e.g., [Kao06], [Ke08], [Yu08], [Lie08] and [Bou09]) or OPNET (e.g., [Kle09]). In [Kao06], the original simulated environment in Evalvid, which simply consisted of an error model to generate corrupted or missing packets, was modified by integrating the transmission and reception modules into NS-2 through adapted UDP-based agents. Accordingly, the resulting toolset allowed the assessment of various designs and proposals for video streaming over heterogeneous network scenarios. In [Ke08], that framework was adapted by incorporating a Multiple Description

Coding (MDC) technique, which was evaluated over a wireless scenario. These previous platforms were only focused on video quality evaluation, so an extended toolset that also supported the evaluation of the audio quality was presented in [Yu08]. In [Lie08], the combined toolset using Evalvid and NS-2 was enhanced by integrating a solution for rate adaptive MPEG-4 video streaming. In [Bou09], the previous work was adapted to be used in multicast scenarios. In [Kle09], Evalvid was integrated into OPNET simulator, and the resultant framework was used to evaluate the performance of several routing algorithms in video streaming applications over multi-hop wireless ad-hoc networks.

Unlike the previous platforms in which the QoS metrics are mostly evaluated at the end of the simulation process, our toolset takes advantage of the RTP/RTCP capabilities to enable the measurement of QoS metrics during simulation time. For instance, the feedback information provided by RTCP packets could enable researchers and practitioners to assess their novel designs (such as network protocols, routing strategies or coding mechanisms) for video streaming applications in heterogeneous network scenarios, under different conditions.

Figure 10.6 illustrates the structure of the designed toolset for video quality evaluation, the interactions between the different involved tools and the necessary steps to perform a video streaming test. The toolset has been constituted by combining NS-2, including our new developed RTP/RTCP module, video encoders/decoders (such as *ffmpeg*[40]), new adapted programs to generate Traffic Trace Files from video files and vice versa, VLC media player, a YUV Viewer[41], and additional programs to calculate application-level video quality metrics, such as Video Quality Measurement Tool (VQMT) [42].

In order to perform a video streaming evaluation using this toolset, three phases must be followed, namely *pre-processing phase* (it is sketched in the upper left corner of Figure 10.6), *simulation phase* (lower part of Figure 10.6) and post-*processing phase* (upper right corner of Figure 10.6). The *pre-processing phase* consists of encoding a video file from a given raw file, probably in YUV format, using, for example, *ffmpeg* tool. Otherwise, if an encoded video file is already available, it can be decoded to YUV format, by also using *ffmpeg* tool, in order to generate a reference video for quality measurement during the post-processing phase. During this phase, different encoding mechanisms (e.g., MPEG-4, H.263, H.264…) can be chosen. This enables a possible comparison between their efficiency and suitability for the specific video applications and scenarios. Moreover, various encoding parameters, such as the frame rate, bit rate, quantizer

[40] FFMPEG program, http://sourceforge.net/projects/ffmpeg/.

[41] YUV viewer, http://www.brothersoft.com/elecard-yuv-viewer-download-142207.html.

[42] MSU Video Quality Measurement Tool (VQMT). Graphics&Media Lab, Moscow State University, http://compression.ru/video/quality_measure/video_measurement_tool_en.html.

scale (Q) or GoP size and pattern, can also be configured. The tuning of such parameters can be very useful for adjusting the transmission rate according to the available resources (bandwidth, memory, CPU, etc.) or the video quality required for specific applications.

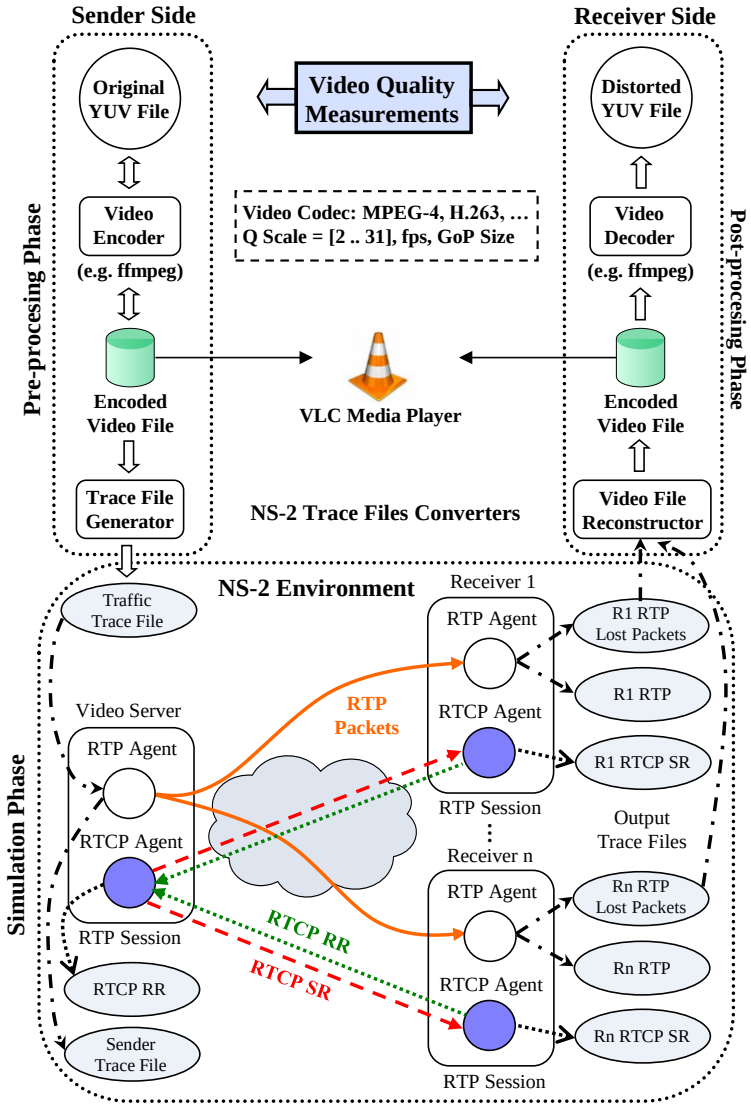


Figure 10.6. Toolset Structure.

After that, the encoded video file is converted, using a *Video Trace File Generator*, into an input trace file to the developed RTP Agent. This input trace file will be an abstraction of the real video stream to be sent, containing useful information, such as the frame number, frame type, frame size and generation time for each one of them.

Next, the *simulation phase* can be initiated. First of all, the generated Video Trace File has to be attached to the RTP Agent for the video server node of the simulated scenario. Therefore, once the simulation is started, the sender RTP Agent will read the information from this trace file, and start sending RTP packets to the targeted receiver/s, also based on the allocated bandwidth for the RTP Session, the unicast or multicast nature of the session, and the configured value for the MSS. The sender RTP Agent will also generate output trace files including the sequence number, timestamp, size and wall-clock transmission time of each RTP packet. Likewise, the wall-clock reception time, sequence number, timestamp, delay, jitter, and size for each received RTP packet will also be recorded in output trace files at each one of the video receivers. Moreover, the total number of RTP packets lost and the average throughput are also recorded. Similarly, the statistics from the RTCP packets are also recorded in output trace files at both the sender and receiver sides.

Moreover, a decodable frame rate module has also been included at the receiver side. This module allows predicting, during simulation, the degradation of the video quality due to loss or late arrival of video frames, identifying the type of missing frame (e.g., I, P or B frame).

When the simulation is over, the RTP statistics recorded in the output RTP trace files can be used to reconstruct the transmitted video files at the receiver side, by using the developed Video File Generator program. If the output trace files contain information about lost packets, the entire video frames to which these packets belong are considered lost, since they could not be correctly decoded. Thus, this program has to insert all missing frames due to drops (or late arrivals) so that sent and received videos consist of equal number of frames, which is required for calculating the PSNR. This process is performed by copying the last successfully decoded frame to each frame that has been lost, as a simple error concealment technique. At this point, the reconstructed (possibly distorted) video files can already be played by a media player, such as VLC. However, they must also be decoded, e.g. using *ffmpeg* tool, to a raw file in order to use it for evaluating the quality of the end-to-end video (e.g., using VQMT program). As the reconstructed video files at receiver side can be played out, using either VLC or YUV viewer program, subjective assessments can also be conducted.

A brief discussion on common video quality metrics that can be measured using this tool-set and their correlation with the human perception is provided in [Bor11a].

More details about the implementation of this toolset and its use for the assessment of QoS metrics in a simple video streaming application can be found in [Bor11a].

10.4 Prototype in a Real Media Framework (GStreamer)

This Section describes the prototype implementation in GStreamer including various components of the developed IDMS solution. First, an introduction to GStreamer is provided in Section 10.4.1. After that, the different GStreamer modules that implement the RTP/RTCP and jitter buffer functionalities are presented. In Section 10.4.3, the required GStreamer elements to constitute the transmission and reception pipelines are introduced. Then, the two mechanisms that can be used for clock synchronization in GStreamer are presented in Section 10.4.4. Next, Section 10.4.5 describes the integration of the previous GStreamer components with the RTSP and SDP modules. Finally, the developed methods to automatically and visually measure the end-to-end delay and IDMS performance are presented in Section 10.4.6.

10.4.1 Introduction to GStreamer

GStreamer is a powerful and versatile open-source framework for creating multimedia applications that handle audio, video, or any kind of data. It is cross-platform supported (e.g., it can be installed on Windows, Linux, Android, Mac and iOS) and provides bindings to several programming languages (such as Python, Java and C).

GStreamer is based on plugins, where each plugin contains the required elements to perform specific tasks, such as reading/writing from/to files, encoding, decoding, filtering or rendering data. The available plugins (or elements) can be linked and arranged in pipelines in order to develop diverse full-fledged multimedia applications.

Elements are the basic building blocks of GStreamer and can be classified into the following categories:

- Source elements: Generators of data, such as audio, video or file sources.
- Filter elements: Elements that transform or process data, such as encoders, decoders or volume control.
- Sink elements: Destinations of data, such as audio and video sinks.

Figure 10.7 shows an example of a simple pipeline composed of three linked elements. The media data flows downstream from a Source element, through one or more Filter elements until reaching a Sink element. The first element in a pipeline is always a Source element, which generates and/or inserts the data. Likewise, the

last element in a pipeline is always a Sink element, which renders the data to the targeted destination (displays, loudspeakers, files…). The output of a Source element will be used as input to a Filter (or Sink) element. The Filter element will process the data and forward them to the next element in the pipeline. Elements are connected via Pads, which represent the "plugs" or "ports" on elements where links may be made between elements, and through which data can flow to or from those elements. Data flows out from one element through one or more source (*src*) pads, and elements accept incoming data through one or more sink pads. Source and sink elements have only source and sink pads, respectively. The data is contained into buffers, which are the basic units of data transfer in GStreamer and can include one or more frames.

Likewise, the GStreamer plugins can be mainly classified into the following categories:

- Protocols handling.
- Media (audio and video) sources.
- Formats: parsers, formaters, muxers, demuxers, metadata, subtitles, etc.
- Codecs: coders and decoders.
- Filters: converters, mixers, effects, buffers, etc.
- Media (audio and video) sinks.

Figure 10.8 shows an overview of the GStreamer architecture. The core framework provides the infrastructure for the system (e.g., pipeline architecture, plugin system, media handling, base classes, communication bus…) and exposes a set of interfaces to integrate and/or inter-operate with other (probably third-party) components (e.g., tools, applications or even systems). Moreover, the existing GStreamer components can be modified and new components can be developed thanks to its modular design and open-source nature. A complete documentation about the GStreamer framework can be found in http://gstreamer.freedesktop.org/documentation/.

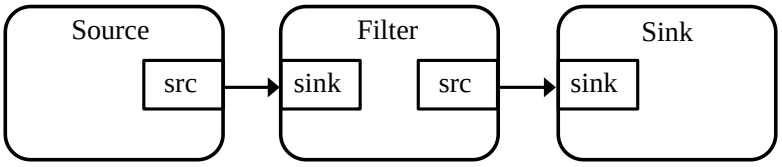


Figure 10.7. Example of a Simple GStreamer Pipeline.

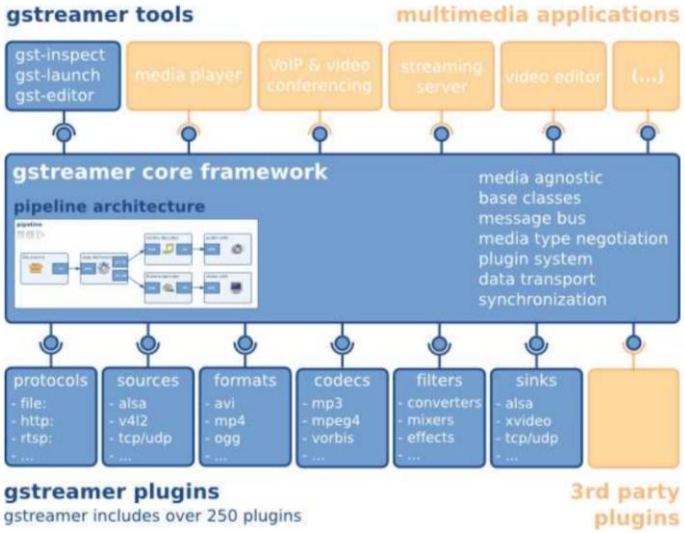


Figure 10.8. GStreamer Architecture Overview: Core Framework, Plugins, Tools and Applications.

10.4.2 RTP/RTCP Support in GStreamer

GStreamer provides an accurate and complete support of the RTP/RTCP functionalities. Such functionalities are mostly implemented in the *rtpbin* component (a bin is a container for a collection of elements in GStreamer), which belongs to the *rtpmanager* plugin, and is composed of the following main elements or modules (see the interactions between the most relevant ones in Figure 10.9):

- RTP Session (implemented in *rtpsession* component): It is mainly responsible of sending and receiving RTP and RTCP packets, as well as maintaining the participants' statistics.
- RTP PT Demux (implemented in *rtpptdemux* component): It acts as a demuxer for RTP packets based on the PT value of the incoming RTP packets. Its main purpose is to allow an application to properly process an RTP stream with multiple PTs, by configuring the required elements and properties to process such RTP stream.
- RTP SSRC Demux (implemented in *rtpssrcdemux* component): It acts as a demuxer for RTP packets based on the SSRC value of the incoming RTP packets. Its main purpose is to allow an application to properly process an RTP stream with multiple SSRCs, by configuring the required elements and properties to process such RTP stream.

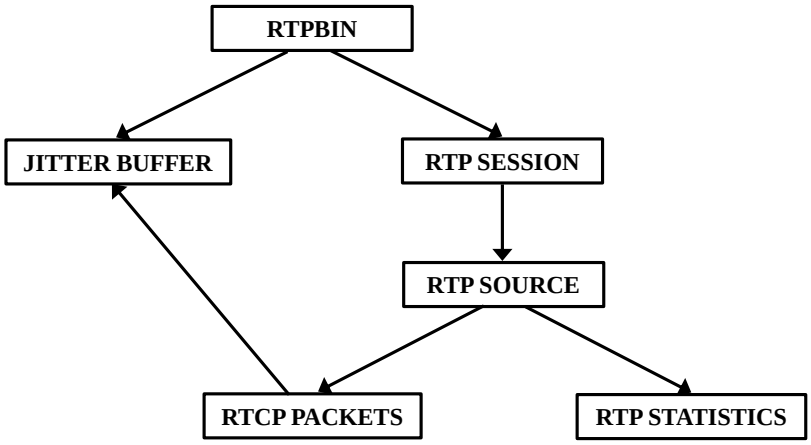


Figure 10.9. Relevant components of *rtpbin* in the developed GStreamer prototype.

- RTP Jitter Buffer (implemented in *rtpjitterbuffer* component): It is a buffer that deals with network jitter, out-of-order packets, duplicates and packet losses. It will wait for missing RTP packets up to a configurable time limit using the *"latency"* property. Packets arriving too late are considered to be lost. The *rtpjitterbuffer* is also used to adjust clock drifts between senders and receivers. It uses the Decoding Timestamps (DTS) and Presentation Timestamps (PTS) of the incoming buffers, as well as the RTP timestamps (in both RTP and RTCP packets) and NTP-based timestamps (in RTCP packets), to adjust the timestamps of the outgoing media data to be played out.

The *rtpbin* component allows the configuration of multiple RTP sessions to be synchronized using the information (timestamps and participants' identifiers) included in RTCP packets. Therefore, the *rtpbin* component enables intra-media and inter-media synchronization by relying on the RTP/RTCP functionalities (correlation between RTP and NTP timestamps and between SSRC and CNAME identifiers) and on a proper configuration of the *rtpjitterbuffer*.

10.4.3 Transmission and Reception Pipelines

The sub-section briefly introduces the main elements that compose the transmission and reception pipelines of the developed GStreamer prototype (see Figure 10.10):

- Video Source element: It is the input of the pipeline. It can read, for example, from a stored media file (e.g., *filesrc* element) or from a live capturing webcam (e.g., *v4l2src* element).

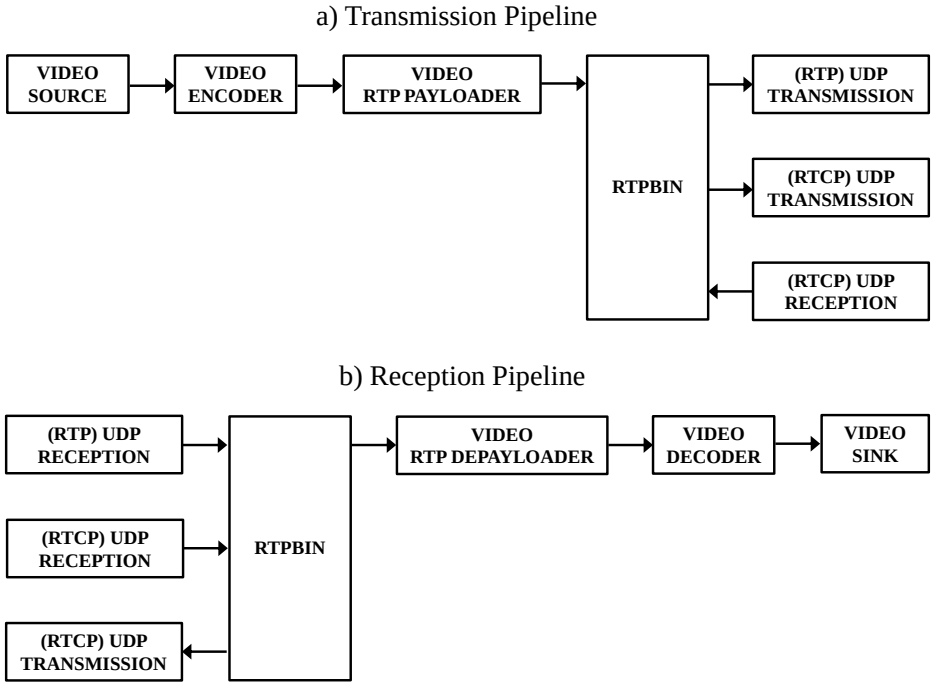


Figure 10.10. Transmission and reception pipelines in the GStreamer prototype.

- Video Encoder element: It encodes the raw video frames (e.g., in YUV format) using a specific video codec (e.g., *x264enc* for H264 codec, or *theoraenc* for Theora codec).
- Video Decoder element: It decodes video frames into raw frames (e.g., *avdec_h264* for H264 codec, or *theoradec* for THEORA codec).
- Video Payloader element: It packetizes encoded video frames into RTP packets (e.g., *rtph264pay* for H264, or *rtptheorapay* for Theora).
- Video Depayloader element: It extracts (i.e., de-packetizes or reconstructs) encoded video frames from RTP packets (e.g., *rtph264depay* for H264, or *rtptheorapay* for Theora).
- Video Sink element: Video output element (e.g., *xvimagesink* for a XV based sink).
- UDP Transmission element: It sends data over the network via UDP (using *udpsink* element), via unicast or multicast, using a specific port.
- UDP Reception element: It receives data over the network via UDP (using *udpsrc* element), via unicast or multicast, using a specific port.

Although only the components for video streaming have been presented, the equivalent components for audio streaming are also used to provide audio support. Likewise, other video streams could also be added to the pipelines.

Table 10.1 summarizes the most relevant properties of the involved components that are useful to provide multimedia synchronization in the developed GStreamer prototype.

Table 10.1. Attributes for multimedia synchronization control in the GStreamer prototype

GStreamer Component	Property	Purpose
rtpbin	*"ntp-sync"*	Set the NTP time from the RTCP SRs as the *running-time* on the buffers.
rtpbin	*"use-pipeline-clock"*	Use the pipeline *running-time* to set the NTP time in the RTCP SR messages.
rtpbin	*"sdes"*	Set the SDES items in the RTCP messages.
rtpbin	*"rtcp-sync"*	Use of RTCP SR for synchronization.
rtpbin (rtpjitterbuffer)	"latency"	Maximum latency the RTP packets will be kept in the jitter buffer.
rtpbin (rtpjitterbuffer)	*"buffer-mode"*	Set the buffering algorithm in use in the jitter buffer.
rtpbin (rtpjitterbuffer)	*"drop-on-latency"*	Drop oldest buffers when the buffer is completely filled.
rtpjitterbuffer	*"ts-offset"*	Adjust the output timestamps of the buffers according to this value (in ns). This is useful to trigger synchronization (i.e., playout) adjustments.
sink	*"ts-offset"*	Adjust the output timestamps of the buffers according to this value (in ns).
sink	*"render-delay"*	Set an additional delay between synchronization and actual rendering of the media. This property will add additional latency to the device in order to make other sinks compensate for the delay.
sink	*"slave-method"*	Set the specific playout adjustment strategy (compensation algorithm) to synchronize with the "master" clock.
sink	*"drift-tolerance"*	Allowed asynchrony threshold between the clock of the sink element and the master ("shared") clock.
sink	*"sync"*	Synchronize on the "master" clock of the pipeline.
sink	*"provide-clock"*	Provide a clock to be used as the global pipeline clock.

10.4.4 Clock Synchronization

The developed prototype can make use of two mechanisms to achieve clock synchronization (see red boxes in Figure 10.11 and Figure 10.12). The first one is based on synchronizing the clocks of all participants via NTP (e.g., using *ntpdate* program or other NTP clients) and then properly configuring the *"ntp-sync"* and *"use-pipeline-clock"* properties of *rtpbin* component (see Table 10.1). The second one consists of using two GStreamer components: i) *NetTimeProvider*, which exposes a "master" clock on the network; ii) *NetClientClock*, which subscribes and gets enslaved to that "master" clock and periodically polls for its values.

10.4.5 Integration with RTSP and SDP Modules

The RTP/RTCP and clock synchronization modules have been also integrated with the SDP and RTSP modules of GStreamer. On the one hand, the SDP module enables the negotiation and description of the parameters in use for the media session. On the other hand, the integration with the RTSP functionalities allows for configuring an RTSP server, with various URL mount points, each one allocating either stored or live media content, which can be sent to distributed RTSP clients in either a unicast or multicast way. A pool of multicast addresses (IP addresses and ports) can be also allocated for delivering (or sharing) the different media resources.

The RTSP server and client/s architectures are shown in Figure 10.11 and 10.12, respectively.

The standard-compliant behavior of the developed GStreamer prototype has been checked by analyzing the audio and video streams in Wireshark and by playing the media using other media frameworks, such as VLC and MPlayer.

10.4.6 End-to-end delay measurements

Apart of the RTP-to-RTP interfaces delay per each RTP packet and the RTT per each RTCP interval, the developed prototype is able to measure the capture-to-render delay for each video frame in customizable media streaming scenarios. Unlike other existing delay measurement systems (e.g., [Jan13a], [Kry13], [Koo14]), the developed method does not require any users' involvement and it is fully integrated into the media framework, in which a full control on all involved components is available. It is not focused on measuring delays on proprietary closed systems (*"black boxes"*).

In order to enable automatic capture-to-render video delay measurements, two GStreamer elements are used. At the server side, an element called *videomark* is used. It allows overlaying a barcode into each video frame. This barcode can include a 64-bit integer value, which in our prototype represents the capturing NTP-based timestamp. This element is placed just after capturing/retrieving each frame (see blue box in Figure 10.11). At the client side, an element called *videodetect* is used.

It is able to detect changes in a specific video pattern and will be responsible of decoding the timestamp inserted into the barcode. This element is placed just before rendering each video frame to the display (see blue box in Figure 10.12). This way, by comparing the capturing and rendering NTP-based timestamps, the developed prototype is able of automatically measuring (and logging) the capture-to-render delay for each incoming video frame. Obviously, the accuracy of the delay measurement system relies on the accuracy on the employed technology for clock synchronization (e.g., NTP or NetTimeProvider/NetClientClock).

Note that for a complete end-to-end delay (for video, it is also commonly known as "glass-to-glass" delay) the capturing and rendering delays must also be considered. However, the magnitudes of these delays can be neglected (a few ms) compared to the total end-to-end delay [Jan13a, Koo14], and will be roughly the same in homogeneous devices. These sources of delay can be considered by using external tools, such as *videoLat* [Jan13a]. Also, note that we are not considering the end-to-end delay for audio, but that accurate inter-media synchronization mechanisms are supported to adjust any differences between audio and video delays.

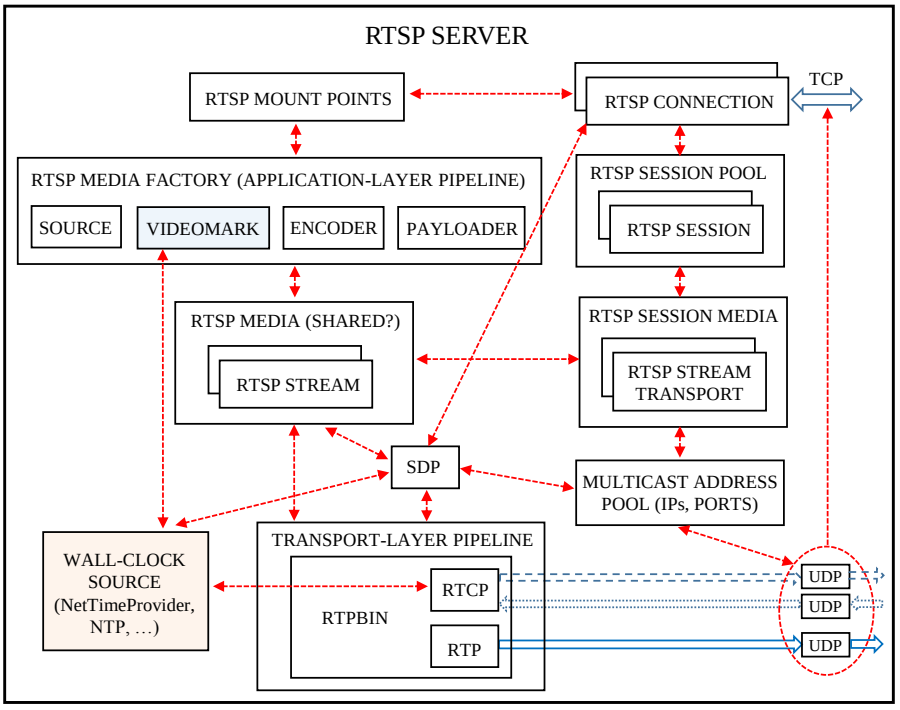


Figure 10.11. RTSP Server Architecture.

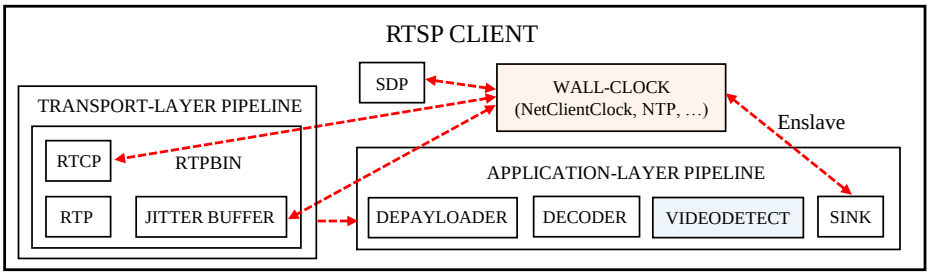


Figure 10.12. RTSP Client Architecture.

Additionally, the developed prototype allows for visually checking the end-to-end delay and IDMS performance. This is achieved by overlaying numeric timestamps and frame numbers for each captured/rendered video frame, by using a customization of a GStreamer element, called *timeoverlay* (see Figure 10.13.) Moreover, the delay and IDMS performance can be measured by launching snapshots, either in a user-transparent way when specific internal conditions are met (e.g., based on the output of the RTP jitter buffer, frame number or system time) or in a manual way by pressing a button (see Figure 10.14).

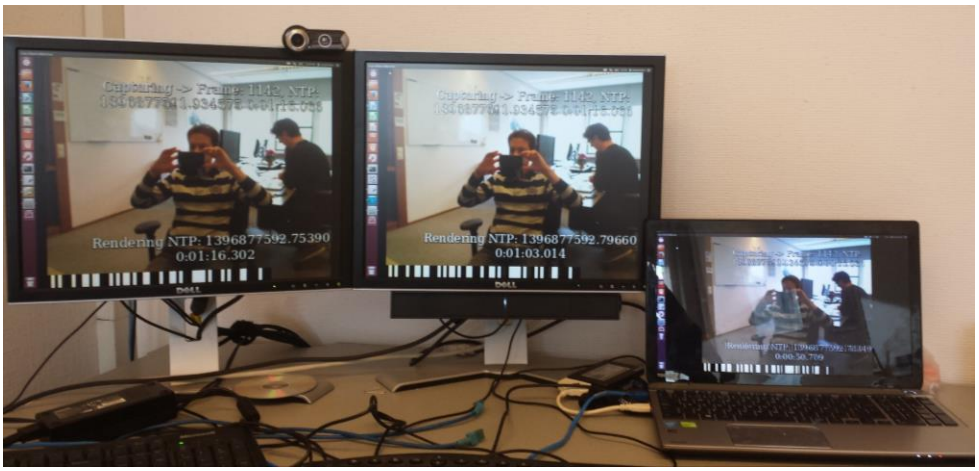


Figure 10.13. Synchronized Playback across Devices, Time Stamped Barcodes and End-to-End Delay Measurement.

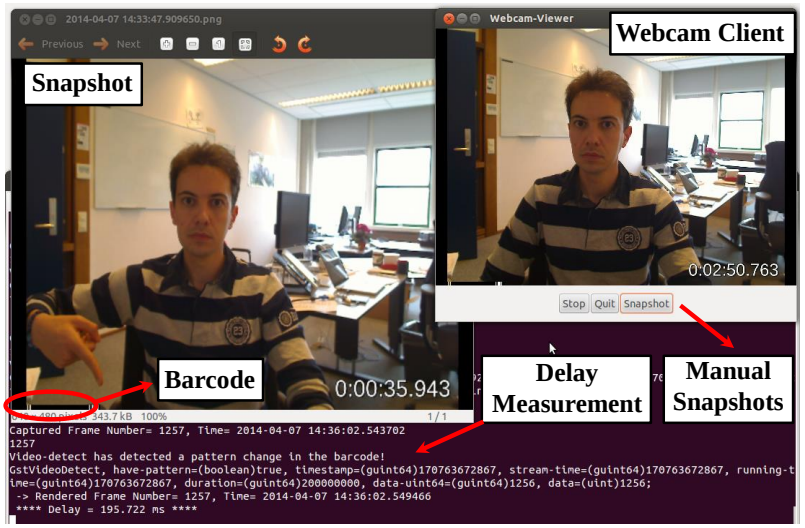


Figure 10.14. Time Stamped Barcode, Snapshots Launching and End-to-End Delay Measurement.

10.5 Summary

In this Chapter, the evaluation methodology and the prototypes that have been implemented in this PhD thesis have been presented.

On the one hand, a new NS-2 module with a complete and accurate implementation of RTP/RTCP has been developed. This module has been extended with an optional functionality including all the components, with their proposed alternatives, of the designed IDMS solution. Moreover, this NS-2 module can also be integrated with other multimedia tools to constitute a powerful and realistic toolset for video streaming evaluation. The source code, installation guides and the necessary steps for performing a simple evaluation are available at our website.

On the other hand, another prototype using GStreamer has also been developed. It does not include all the components of our IDMS solution yet, but the essential ones to show its feasibility and performance in real scenarios. The final prototype implementation in GStreamer will serve as testbed to exhaustively assess the QoE in different use cases, under different situations. However, this is out of the scope of this PhD thesis, which is mostly focused on technical issues and, therefore, on an objective evaluation. Demo videos can be watched at http://goo.gl/bK50i7, and at http://goo.gl/xcMF05.

As well, the GStreamer code will be available to the scientific community after our tests are finished.

Chapter 11

EVALUATION IN NS-2

11.1 Introduction

This Chapter presents the evaluation of our RTP/RTCP-based IDMS solution using NS-2. First, the simulation scenario and setup are described in Section 11.2. Then, the evaluation of each one of the components of the IDMS solution is provided in the subsequent Sections.

11.2 Simulation Setup and Scenario

The designed IDMS solution has been tested in the NS-2 multicast scenario shown in Figure 11.1. This scenario has seven distributed Sync Clients, with variable network delays to the Media Server (see Table 11.1), belonging to two different logical synchronization groups (Group 1 -G1- and Group 2 -G2-). All the links were bidirectional, their propagation delays were set to 10 ms, and their capacity was configured as shown in the figure. The Media Server transmitted a stream with a specific rate of $\theta=25$ *MU/s* (i.e., constant MU rate). When using SMS, the Sync Manager was co-located with the Media Server.

In addition, several aspects were considered to originate significant end-to-end delay variability between the involved sync entities. First, apart from the RTP/RTCP traffic, heavy and fluctuating background traffic (concretely, different cross-traffic flows following CBR over UDP, FTP (File Transfer Protocol) over TCP, and Pareto over UDP patterns) was configured over the network topology to force significant jitter variability. The intensity of the background traffic was set in order to assure that the total amount of network traffic (RTP/RTCP + background traffic) was near the links' capacity at some instants during the simulations. Second, the Sync Clients were strategically placed such that significant network delay variability from the Media Server to each one of them exists (see Figure 11.1). This differs from the evaluation in [Bor09c], in which the Sync Clients were placed in two different LANs

(called local and remote clusters). However, the Sync Clients belonging to G2 were placed close together (but far from the Media Server) to show the benefits of DCS, compared to SMS, in such a case. Third, significant playout rate deviations (skews and drifts) were configured in the Sync Clients' playout processes (see Table 11.1). These values were set larger than customary deviations in inexpensive oscillators, which can vary between 10-100 *ppm* [Fer10], in order to force higher asynchronies between the involved Sync Clients, and to test if such asynchronies could be successfully handled by our IDMS solution. Moreover, in order to corroborate the M/S switching capabilities of our IDMS solution (when using SMS and DCS), those values were intentionally changed in two of the Sync Clients belonging to G1 at the midpoint of the simulation (*300-th* second), as reflected in Table 11.1. Fourth, an optional functionality was included to simulate congestion situations at the Sync Client side (e.g., due to CPU overload or processing delays), which can cause abrupt discontinuities in the media playout and, therefore, a critical loss of synchronization. This congestion module ran on top of the local playout processes of each Sync Client, and it was configured by adopting an Exponential ON/OFF distribution, in which the active period (ON) implies severe congestion situations, and the inactive period (OFF) implies no congestion cases (i.e., normal playout process). This way, every time the active period is triggered, the MU being played out at that moment is stretched until this congestion period ends, causing a probable *freezing effect*. The congestion module was enabled in some of the simulations to Sync Client 2 (in G1), with t_{ON}*=40 ms and* t_{OFF}*=120 s*, in order to force higher playout times discrepancies between the Sync Clients in that group.

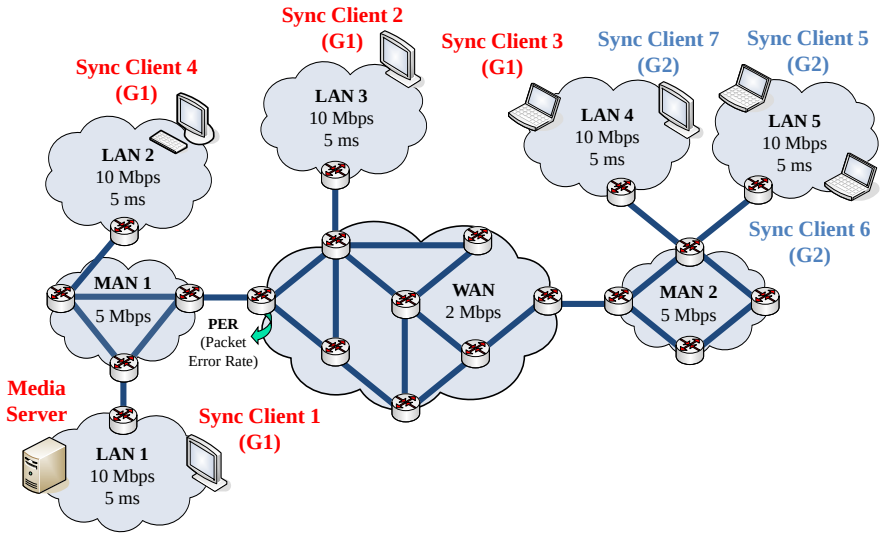


Figure 11.1. Simulated Scenario.

Table 11.1. Sync Clients' Parameters and Aggrupation

Sync Client (SC)	Group	Mean RTT (ms)	Playout Rate Skew, γ (%)	Playout Rate Drift, ε (%)
SC1	G1	~10	0.03 %	0.02 %
SC2	G1	~125	- 0.02 % → - 0.03 %	0.02 %
SC3	G1	~288	- 0.05 % → - 0.02 %	0.02 %
SC4	G1	~44	- 0.015 %	0.02 %
SC5	G2	~288	0 %	0.02 %
SC6	G2	~288	- 0.02 %	0.02 %
SC7	G2	~288	0.01 %	0.02 %

With respect to wall-clock synchronization, the involved sync entities made use of the global scheduler clock of the simulator as the absolute shared, as well as local, timing reference or clock source.

The duration of each simulation was set to 10 minutes and the value of τ_{max} (allowable asynchrony threshold) was set to 80 ms in order to trigger playout corrections slightly before reaching an asynchrony of 100 ms, which can be already perceivable and annoying in many IDMS use cases, as discussed in Section 2.8.

The objectives of the simulation tests were to examine the proper behavior and performance of each of the components of the designed IDMS solution in this PhD thesis. In particular, we mainly wanted to: i) test the correct exchange of the newly designed RTCP messages for IDMS, for each one of the three deployed control schemes (SMS, DCS, and M/S Scheme); ii) check the proper performance of the control algorithms for IDMS (e.g., asynchrony calculation, fault tolerance, coherence…); iii) examine the feasibility and suitability of the different master selection policies for IDMS; iv) assess the accuracy and suitability of the playout adjustment techniques; v) compare the performance of the control schemes (SMS, DCS and M/S Scheme) in terms of several factors, such as interactivity, coherence, traffic overhead and computational load; vi) check the benefits of the proposed EED RTCP Feedback for IDMS compared with Regular RTCP Feedback; etc.

11.3 Intra-Media Synchronization

Figure 11.2 illustrates the buffering and playout delays for one of the Sync Clients (SC3 in G1), when there were not playout rate deviations ($\gamma_1=0$, $\varepsilon_1=0$), in two different network load cases. Initially, we ran a single simulation without background traffic. In that case, the traffic in the network was mainly given by the RTP data stream and the associated RTCP feedback packets (without considering the control messages associated with network management and the IP multicast processes). As a result, both the buffering ($b_3 \approx p_3 - l_3 \approx 500-(288/2) \approx 500-144 \approx 356$

ms)[43] and playout ($p_3 \approx 500$ *ms*) delays for that Sync Client were quite stable during the multimedia session, because the jitter for the multimedia traffic was insignificant. This can be seen in the red and blue graphs in Figure 11.2. In the next simulation, the network load was increased by transmitting intensive background traffic. As expected, when increasing the total traffic load, the buffering delay for that Sync Client presented significant fluctuations during the session (see green graph in Figure 11.2), mainly due to the higher queueing delays at the intermediate routers (as RTP traffic must contend with background traffic). Accordingly, the jitter values for the received RTP media stream were much higher and more variable than in the previous case. However, in both network load cases, the designed playout buffering policy was able to smooth out the effect of the time-variant jitter, by delaying the incoming MUs at the playout buffer, and then removing them from the buffer queue with the same rate as they were sent by the Media Server. Therefore, the original media timing for the incoming RTP stream was reconstructed in both cases, resulting in a uniform playout delay, as can be seen in the blue graph in Figure 11.2 (high quality of intra-media synchronization).

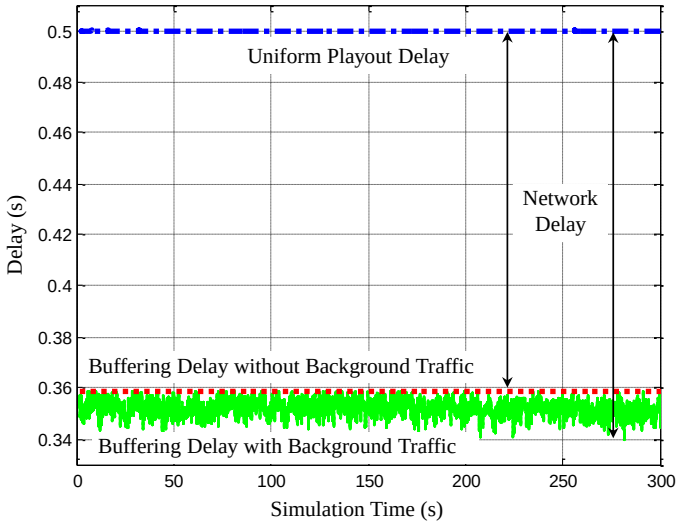


Figure 11.2. Intra-Media Synchronization (Uniform Playout Delay).

The effect of the playout rate imperfections is shown in Figure 11.3. This figure illustrates the playout delay evolution for the same Sync Client, when different rate deviations were configured to its playout process in the simulation setup. First, it

[43] As shown in Table 11.1, the mean RTT for SC3 was 288 ms. Thus, it is assumed that the network delay for SC3 was around $l_3 \approx (288/2) \approx 144$ ms.

can be seen that when that Sync Client presented an ideal playout rate (i.e., a playout rate with $\gamma=0$, and $\varepsilon=0$), the incoming MUs were played out at a constant MU rate, thus presenting a uniform playout delay throughout the duration of the session (see green graph in Figure 11.3). Second, when a playout rate drift was configured to that Sync Client (i.e., $\gamma=0$, and $\varepsilon\neq0$), its playout rate presented random and time-variant oscillations over the nominal playout rate (as described in Section 2.6 and shown in Figure 2.7). This resulted in a slight time-variant fluctuation of the playout delay (see red graph in Figure 11.3). Third, when both a playout rate skew and drift were configured to that Sync Client (i.e., $\gamma\neq0$, and $\varepsilon\neq0$), its playout rate additionally presented a deviation trend over the nominal rate. Depending on the value of the rate skew (γ), the playout rate of the Sync Client will be slower (if $\gamma<0$) or faster (if $\gamma>0$) than the nominal playout rate. As a consequence, its playout buffer may progressively become *flooded* or *emptied* with MUs if such effect is not handled during the multimedia session. In the simulated case, the configured playout rate deviations to that Sync Client were $\gamma=-0.05\%$ and $\varepsilon=0.03\%$. Accordingly, its playout rate was slower than the nominal rate, and the playout delay for that Sync Client progressively increased (see blue graph in Figure 11.3), probably causing a buffer overflow situation if the multimedia session had a long duration.

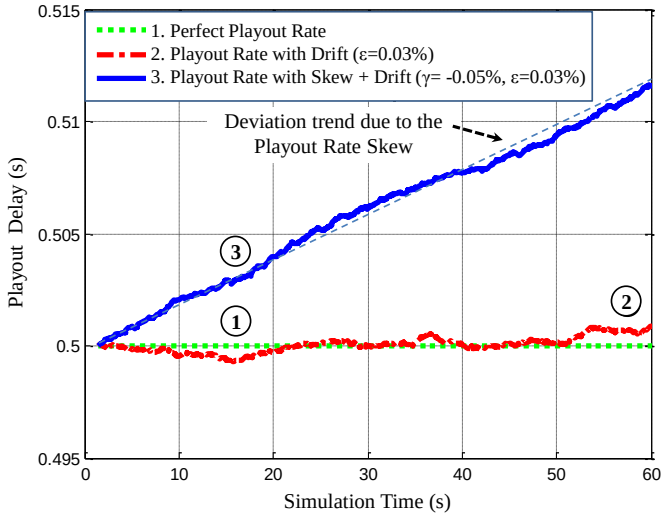


Figure 11.3. Effect of Playout Rate Imperfections on Multimedia Synchronization.

Apart from intra-media synchronization, such playout rate imperfections will have an impact on inter-media synchronization (because of the different rates of the playout processes of the involved media types) and on IDMS (because of the different rates of the playout processes of the involved Sync Clients). In the

simulated case, if we assume that the red graph is the playout process of another Sync Client or of another media type in the same Sync Client, it can also be inferred that the presence of playout rate imperfections (especially of playout rate skews) will result in an unacceptable increasing asynchrony between playout processes that needs to be corrected.

11.4 Evaluation of SMS

In this sub-section, the performance of SMS for IDMS, when using the different master selection policies (described in Section 8.8) and adjustment techniques (described in Section 8.11), is assessed.

11.4.1 Synchronization Policy to the Fastest Sync Client

Figure 11.4 illustrates the playout delay evolution of three Sync Clients belonging to G1 (see Table 11.1), when the IDMS solution was enabled, by using the synchronization policy to the fastest Sync Client, and by using both aggressive adjustments (in Figure 11.4.a) and smooth adjustments (in Figure 11.4.b). First of all, it can be seen in both graphs that all the Sync Clients were perfectly synchronized at the initial playout instant (p_{ini}), because the Sync Manager sent to them an initial RTCP IDMS Settings packet to force a *global initial playout delay* (d_{ini}) of 500 ms, as in the real scenario in [Bor08]. This way, the network delay variability from each of the Sync Clients to the Media Server was initially compensated. Such process can also be appreciated in most of the subsequent graphs presented in this section. Moreover, it can be seen that the asynchrony between the playout processes of the involved Sync Clients progressively increased mainly due to the configured deviations in their local playout rates. In particular, SC1 played out the incoming MUs with the highest rate because it had the highest (positive) skew of all of them, whereas SC3 played out the incoming MUs with the lowest rate because it had the lowest (negative) skew of all of them (see Table 11.1). As a result, every time the Sync Manager detected a playout time discrepancy greater than τ_{max} (80 ms) between the involved Sync Clients, by comparing the timing information from their IDMS reports, it sent an RTCP IDMS Settings packets to notify the Sync Clients in that group the need for adjusting their playout processes. This IDMS Settings packet included a common IDMS target playout point, calculated taking into consideration the collected IDMS timing from SC1, which was the fastest Sync Client ($\gamma_{master}=\gamma_{max}=\gamma_1$) in that group.

On the one hand, when using aggressive adjustments (Figure 11.4.a), slower (slave) Sync Clients had to *'skip'* zero, one or two MUs[44] ($\tau_{max}=2\cdot s_{n,i}=80$ *ms*), as a consequence of the reception of each IDMS Settings packet (see zoom view in

[44] Only entire MUs can be skipped, each one with a duration of $1/\theta$=40 ms/MU.

Figure 11.4.a). However, there were not any *'pauses'* in the playout processes of the Sync Clients during the evolution of the session. On the one hand, when using AMP (Figure 11.4.b), slower (slave) Sync Clients had to smoothly fast up their playout rate to achieve synchronization every time an IDMS Settings packet was received (see zoom view in Figure 11.4.b).

As can be seen in Figure 11.4, lagged Sync Clients were more closely and fine-grained synchronized using AMP than using aggressive adjustments. This is because they were able to minimize the estimated asynchrony by adjusting their playout rate to the most proper value. Using aggressive adjustments, however, a residual asynchrony remained after synchronization in some cases. Therefore, using AMP, the Sync Manager will send a minor number of RTCP IDMS Settings packets in long multimedia sessions.

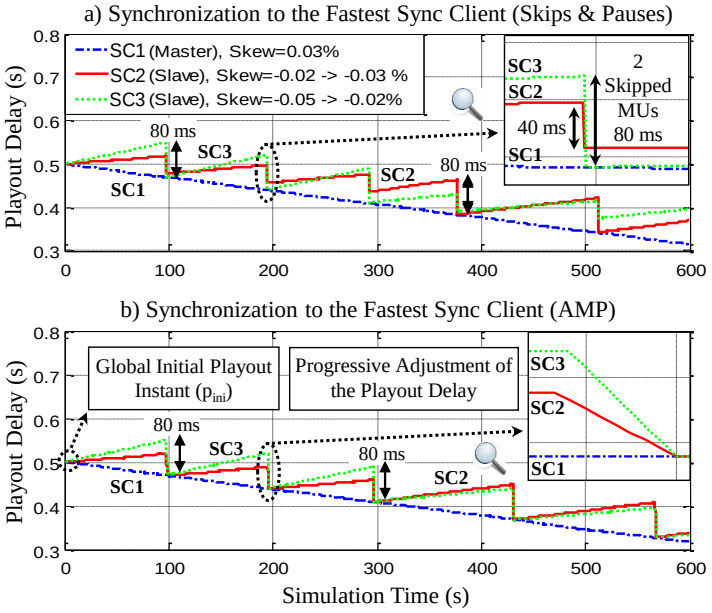


Figure 11.4. Playout Delay Evolution to Achieve IDMS: SMS, using Synchronization Policy to the Fastest Sync Client.

Figure 11.5 represents the same case as Figure 11.4.b, but when six Sync Clients belonging to both G1 and G2 joined the session. This figure shows that our IDMS solution is capable of independently, but concurrently, managing the playout processes of Sync Clients belonging to different groups. In such a case, the fastest Sync Clients in each group were selected as the master synchronization references

(SC1 and SC7, respectively). It can also be appreciated that the synchronization actions (e.g., playout asynchrony monitoring and reactive playout adjustments) were performed separately for each group. The zoom views in that figure show the synchronization at the initial playout instant and the smooth adjustments of the playout delays that were performed in each group every time an asynchrony exceeding τ_{max} was detected.

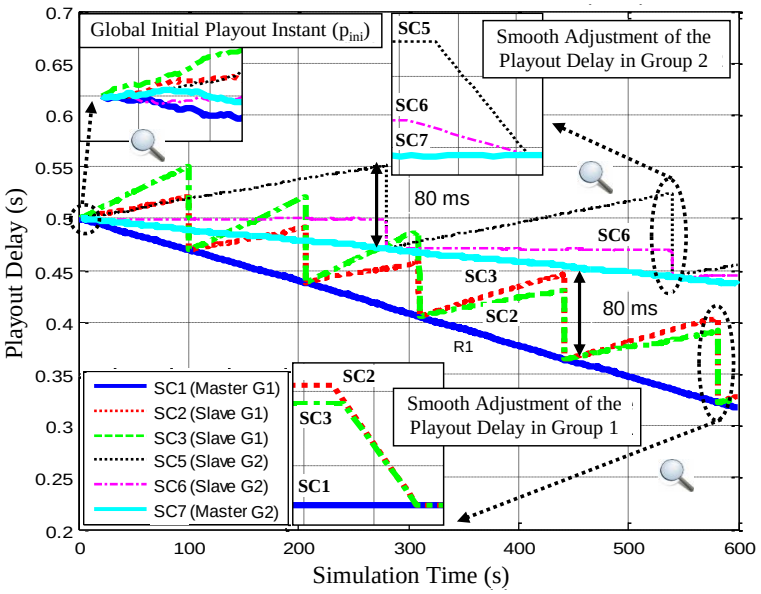


Figure 11.5. Playout Delay Evolution to Achieve IDMS: Group-based SMS, using Synchronization Policy to the Fastest Sync Client.

Figure 11.6 also illustrates the same case as in Figure 11.4.b. However, three additional situations were considered in such a case. First, a new Sync Client (SC4) joined the on-going session at *30-th* second as an additional member of G1. Therefore, the Sync Manager sent a new RTCP IDMS Settings packet to allow that *latecomer* to acquire IDMS, despite that the asynchrony in G1 at that moment was lower than the allowed threshold (see the zoom view at the bottom left). Second, the congestion module was enabled in SC2, therefore causing discontinuities (i.e., disruptions) in its playout process. As a result, the playout asynchrony between the distributed Sync Clients in G1 significantly increased every time SC2 suffered congestion (this can be better appreciated in the zoom views). Third, the session was divided into two stages. Concretely, it was stopped at *300-th* and re-started again at *320-th* second. It can be seen in the zoom views that all the Sync Clients were perfectly synchronized at the beginning of the two stages in which the session was

divided (such process was called *Coarse Synchronization* in [Bor08]), despite of the variable network delays from the Media Server to each one of them, due to the transmission of an initial RTCP IDMS Settings packet by the Sync Manager.

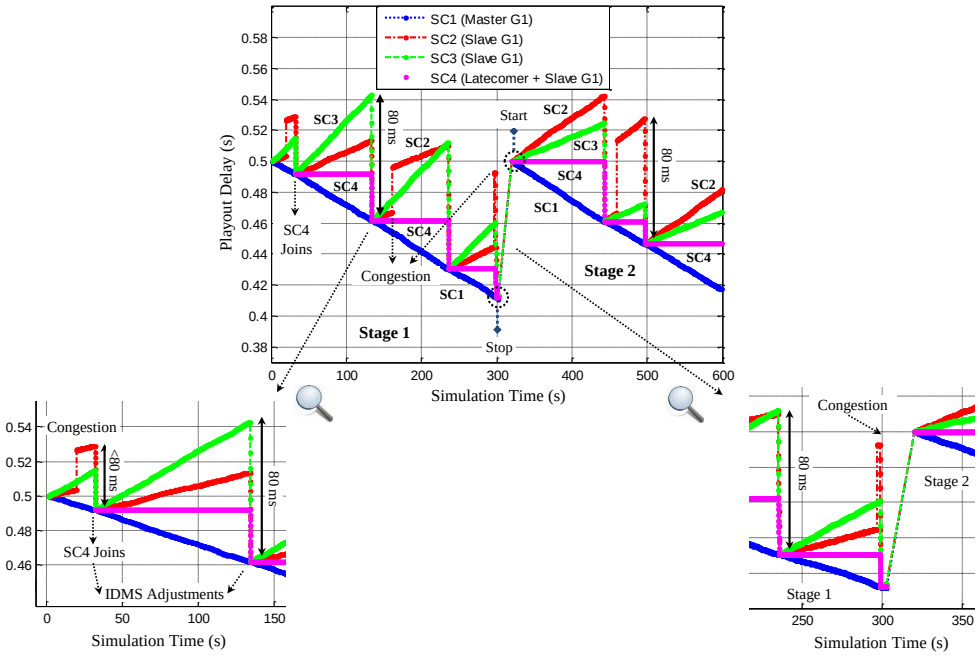


Figure 11.6. Playout Delay Evolution to Achieve IDMS: SMS, using Synchronization Policy to the Fastest Sync Client, Coarse Synchronization, Latecomer's Accommodation and Congestion Situations.

In the previous graphs, in which the synchronization to the fastest Sync Client policy was employed, it can be noticed a significant reduction of the playout delay in all the Sync Clients as the media session advanced in time, which caused an inherent progressive emptying of their buffer occupancy. In Figure 11.5, it can also be noticed a major number of playout adjustments in the Sync Clients belonging to G1, because their playout rate deviation parameters were higher than those of the ones in G2 (Table 11.1). Accordingly, we mainly focus on the measurements for the Sync Clients belonging to G1, by presenting a summary of the statistics of their playout adjustments, for each one of the adopted master reference selection policies. Concretely, Table 11.2 includes the total number and the percentage of adjusted MUs for each Sync Client in G1 when using both aggressive (i.e., the number and percentage of *skipped/paused* MUs, as well as the value of the maximum *'pause'* during the session) and smooth adjustments (i.e., the number and percentage of MUs that were affected by the AMP process, in conjunction with the maximum value of

the playout rate adjustment) during the 10-minute session. Besides, the fourth column of Table 11.2 indicates the maximum variation of the buffer fullness level during the session's lifetime in each analyzed case.

Table 11.2. Summary of Playout Adjustments in Group 1.

Sync Client		Aggressive Adjustments		Smooth Adjustments	
	Master Selection Policy	- Skipped (%) / + Paused (Δ_{max}) MUs	Buffer Fullness Variation (ms)	Number of Adjusted MUs (%)[a]	φ_{max}
SC1	Fastest	0 / 0	- 184.2	-	-
	Slowest	0 / +4 (82.2)	+ 215.8	61 (0.4)	- 0.16
	Mean	0 / + 6 (54.7)	+ 77.9	56 (0.37)	- 0.09
	Source	0 / + 5 (23.7)	$\leq \lvert\tau_{max}\rvert$	43 (0.3)	- 0.08
SC2	Fastest	- 7 (0.05) / 0	-129.8	57 (0.38)	+ 0. 23
	Slowest	0 / + 3 (21.9)	+ 223.8	64 (0.4)	- 0.08
	Mean	0 / + 1 (8.9)	+ 158.2	55 (0.37)	+ 0.06
	Source	- 3 (0.02) / 0	$\leq \lvert\tau_{max}\rvert$	48 (0.32)	+ 0.11
SC3	Fastest	- 8 (0.05) / 0	- 104.9	52 (0.35)	+ 0.24
	Slowest	0 / + 2 (14.5)	+ 235.4	62 (0.4)	- 0.05
	Mean	- 2 (0.01) / 0	+ 127. 8	53 (0.35)	+ 0.1
	Source	- 4 (0.025) / 0	$\leq \lvert\tau_{max}\rvert$	49 (0.33)	+ 0. 12

[a] 14967 MUs were sent during the multimedia session

Using synchronization policy to the fastest Sync Client, we can observe that the buffer fullness level (measured in time, not in MUs) was reduced more than 100 ms for all the Sync Clients (e.g., -184.2 ms for SC1) during the 10-minute session. Note that the maximum reduction is limited by the initial buffering delay (b_{ini}). Therefore, this strategy may not be appropriate if the playout rate of the master is significantly faster than the Media Server nominal rate, because the playout buffers may suffer underflow, and its application would require the use of novel adaptive techniques (e.g. buffer fullness monitoring) or *Coarse Synchonization* actions to avoid such situations, without need of significantly increasing the initial playout delay (d_{ini}) for all the Sync Clients.

11.4.2 Synchronization Policy to the Slowest Sync Client

Figure 11.7 illustrates the playout delay evolution of three Sync Clients belonging to G1 to achieve IDMS when the Sync Manager selected the slowest Sync Client as the synchronization reference ($\gamma_{master}=\gamma_{min}$), by using both aggressive adjustments (in Figure 11.7.a) and smooth adjustments (in Figure 11.7.b). In such a case, faster (slave) Sync Clients had to wait for the slowest one (the master) every time an IDMS Settings packet was received. Unlike in the previous policy, it seems that the graphs

when using aggressive and smooth adjustments are equivalent, as the same synchronization accuracy was achieved when using both types of reactive adjustment techniques. However, it can be seen in the zoom view in Figure 11.7.a that, despite the Sync Clients were accurately synchronized with the IDMS reference, they had to perform long-term pauses (specific MUs were paused a period of time equal to the detected asynchrony). This can be corroborated in the third column of Table 11.2 (e.g., Δ_{max}=*82.2 ms* for SC1). However, when using AMP, the playout delays were adjusted within tolerable ranges to the user perception, as can be confirmed in the last column in Table 11.2. Using this policy, there were not any *'skipped'* MUs during the session.

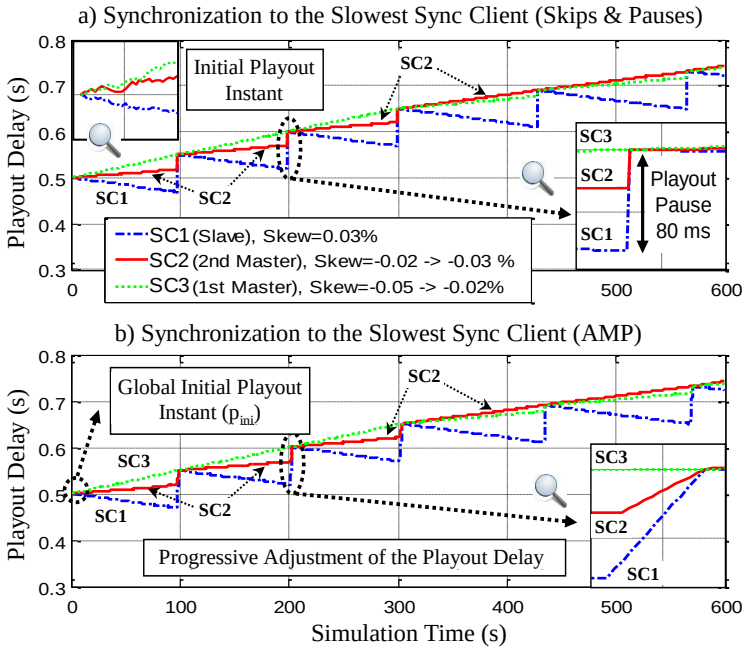


Figure 11.7. Playout Delay Evolution to Achieve IDMS: SMS, using Synchronization Policy to the Slowest Sync Client.

In addition, when using this policy, we can appreciate that the playout delays in all Sync Clients progressively increased as the session advanced in time. This can be corroborated in the fourth column of Table 11.2, which indicates the increase of the buffer occupancy in all the Sync Clients. Therefore, as discussed in Section 8.8, this strategy may not be appropriate if the playout rate of the master is significantly slower than the Media Server nominal rate, because the playout buffers may suffer overflow (which may imply a loss of real-time perception), and its application would

require the use of novel adaptive buffering control techniques to avoid such situations. As an example, in a 90-minute on-line football match session, Coarse Synchronization actions could be triggered at the half time or when the game is stopped by any action (corner, fault…).

As in the previous policy, Figure 11.8 also shows the same situation as in Figure 11.7.b, but when six Sync Clients belonging to both groups joined the session. In such a case, the slowest Sync Clients in each one group (SC3 and SC6, respectively) were selected as the synchronization master references every time τ_{max} was exceeded in their own group (i.e., group-based IDMS). Moreover, this graph clearly reflects the M/S switching capabilities in G1: initially SC3 was the slowest one, but the playout rate deviations of SC2 and SC3 were intentionally changed at *300-th* second (see Table 11.1) in order to force SC2 to become the new master in that group.

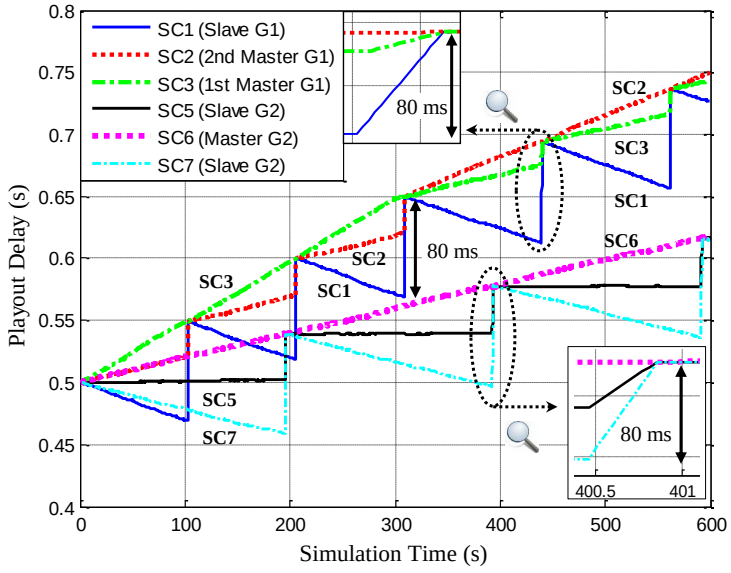


Figure 11.8. Playout Delay Evolution to Achieve IDMS: Group-based SMS, using Synchronization Policy to the Slowest Sync Client.

11.4.3 Synchronization Policy to the Mean Playout Point

Figure 11.9 shows the evolution of the playout processes of three Sync Clients belonging to G1 to achieve IDMS when the Sync Manager selected the Mean Playout Point as the synchronization reference ($\gamma_{master}=\gamma_{mean}$), by using both aggressive adjustments (in Figure 11.9.a) and smooth adjustments (in Figure 11.9.b). This policy reduced the magnitude and frequency of the playout adjustments compared

to the previous ones. For instance, when using aggressive adjustments, SC2 only had to perform a 'short' pause of 8.9 ms during the session, while SC3 only had to perform 2 playout skips, as shown in Table 11.2. When using smooth adjustments, (advanced) lagged Sync Clients smoothly (slowed down) fasted up their playout timing to achieve IDMS, thus avoiding long-term playout discontinuities. However, this solution cannot guarantee buffer overflow or underflow situations because, as discussed in Section 8.8, the existence of extremely advanced or lagged Sync Clients cannot be predicted and would have a quantitative impact on the calculation of IDMS target (mean) playout point. So, as in the previous policies, additional dynamic and adaptive techniques should be adopted.

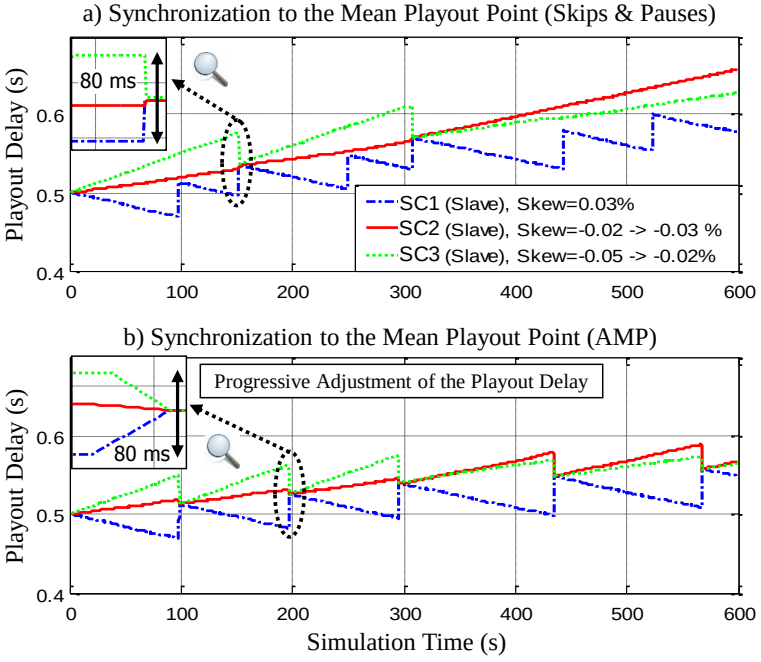


Figure 11.9. Playout Delay Evolution to Achieve IDMS: SMS, using Synchronization Policy to the Mean Playout Point.

As for the previous policies, the group-based SMS operation when using this policy was also tested. This process is shown in Figure 11.10, although the synchronization processes are shown in different graphs for each group for a better clarity.

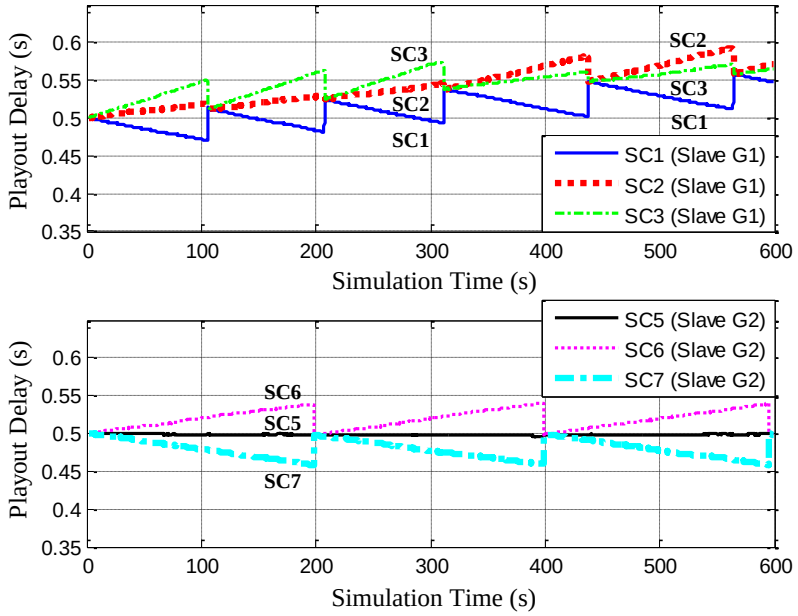


Figure 11.10. Playout Delay Evolution to Achieve IDMS: Group-based SMS, using Synchronization Policy to the Mean Playout Point.

11.4.4 Synchronization Policy to the Media Server Nominal Rate

The synchronization policy to the Media Server nominal rate can minimize the occurrence of buffer underflow and overflow situations, but it is only applicable when using SMS and the Sync Manager being co-located with the Media Server. The upper graph in Figure 11.11 shows the playout delay evolution for the Sync Clients in G1 when using this policy and AMP as reactive adjustment technique. It can be seen that all the Sync Clients adjusted their playout timing to the ideal one indicated by the Sync Manager every time an out-of-sync situation was detected. This resulted in a quite uniform playout delay evolution for all of the Sync Clients, which is a desired feature in real-time multimedia services. Indeed, it can be corroborated in the fourth column of Table 11.2 that when using this policy the variation of the buffer occupancy in all the Sync Clients was bounded to a lower value than $\pm\tau_{max}$ during the session. Thus, starvation situations would be avoided if the network conditions are quite stable. However, this policy does not take into account potential fluctuations of the end-to-end delay (e.g., due to congestion situations). Therefore, as in the previous policies, optimized and adaptive variations of this policy must be devised in future studies to be able to smooth out the effect of jitter and to keep the playout buffers' occupancy into stable and safe ranges.

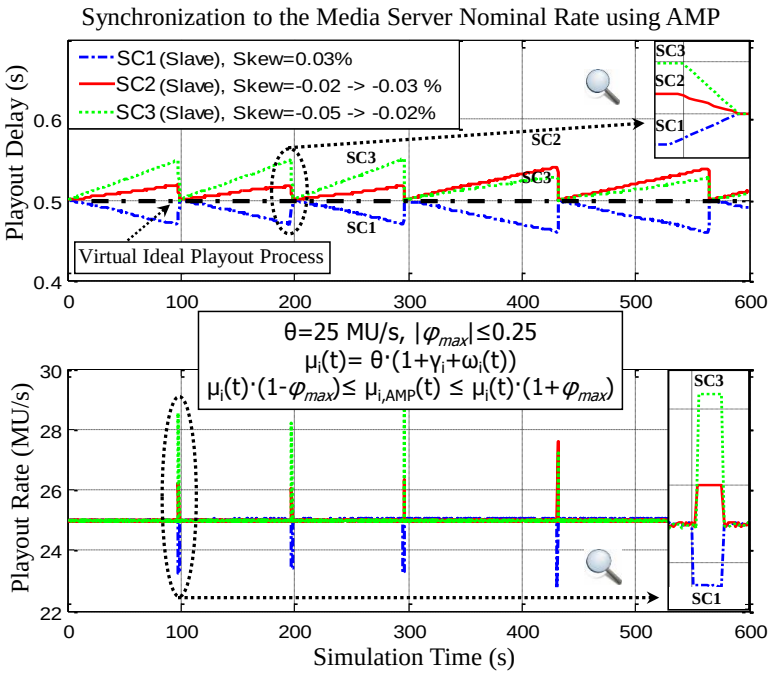


Figure 11.11. Playout Delay Evolution to Achieve IDMS: SMS, using Synchronization Policy to the Media Server Nominal Rate.

The lower graph in Figure 11.11 shows that the playout rate variation in all Sync Clients was varied within perceptually tolerable ranges (i.e., $|\varphi_{max}| \leq 0.25$) in order to achieve IDMS when using this policy. This can also be confirmed in the last column in Table 11.2.

When using aggressive adjustments in this master selection policy, it can be seen in the third column of Table 11.2 that SC1 (fastest) had to perform short pauses ($\Delta_{max}=23.7$ *ms*), and slow Sync Clients (SC2 and SC3) had to *'skip'* a small number of MUs. So, another key advantage of using this policy is that accurate Sync Clients do not have to perform significant playout adjustments, because the IDMS reference is given by an 'ideal' playout timing.

In general, for all the master selection policies, long-term playout discontinuities were avoided when using AMP. However, the total number of adjusted MUs when using AMP was greater than the total number of *skipped/paused* MUs when using aggressive adjustments (see fifth and third column in Table 11.2, respectively). Despite this fact, in none of the master selection policies the percentage of adjusted MUs when using AMP was higher than 0.4 % of the total MUs, which is a very low value that might be unnoticeable by users.

As a summary, Figure 11.12 shows a visual comparative between the (increasing or decreasing) tendency of the playout delay evolution in all Sync Clients when using each one of the master reference selection policies for IDMS.

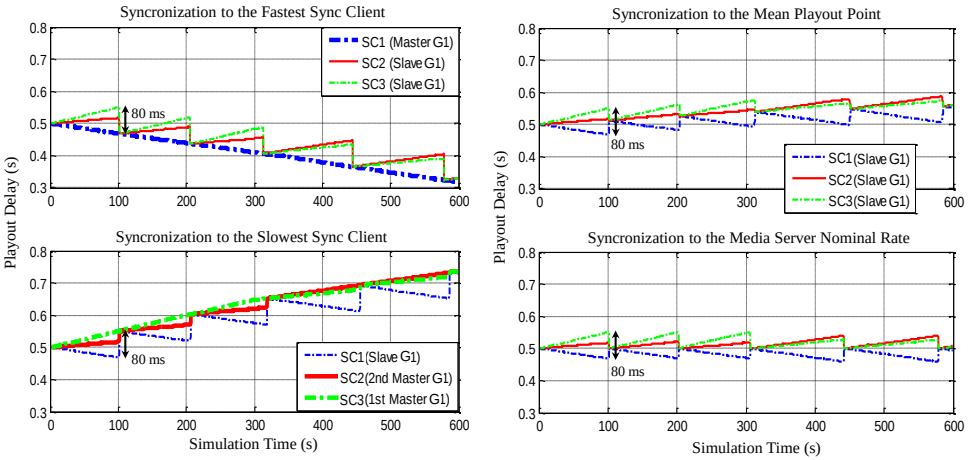


Figure 11.12. Playout Delay Evolution to Achieve IDMS: SMS, Comparison between Master Selection Policies.

11.4.5 Buffer Fullness Variation

To conclude the evaluation of SMS, the same scenario when a new ideal Sync Client with a perfect playout rate (without deviations) joined the multicast session, as an additional member of G1, was tested. The playout controller of that Sync Client was able to play out the incoming MUs with the same rate as they were generated by the Media Server. In that Sync Client, the buffering delay for the incoming MUs was measured when using each one of the master reference selection policies. It can be seen in Figure 11.13 that the buffering delay in that Sync Client was kept quite uniform during the multimedia session (the appreciated fluctuation is due to the jitter delays) until the reception of a new IDMS Settings packet from the Sync Manager. At this moment, that Sync Client had to adjust its playout timing in order to correct the estimated playout time discrepancy, according to the target playout point included in the IDMS Settings packet. As expected, when the synchronization policy to the slowest Sync Client was employed, the buffering delay for the incoming MUs increased after the reception of each IDMS Settings packet. This resulted in an inherent filling of its playout buffer that could even overflow in long multimedia sessions. On the contrary, when the synchronization policy to the fastest Sync Client was applied, the buffering delay in this Sync Client decreased after the reception of each IDMS Settings packet (because $\gamma_{master}=\gamma_1>0$). As a consequence, the buffer

fullness level of that Sync Client was progressive decreasing. For the remaining two policies, a smooth evolution of the buffer delay can be observed, especially when the synchronization to the Media Server nominal rate was employed. So, the buffer fullness level was moderately stable during the multimedia session. However, as previously discussed, although the synchronization policy to the mean playout point minimizes the number and the value of the playout adjustments in all the Sync Clients, this policy is significantly affected by the existence of Sync Clients with considerably deviated playout timings, because the IDMS target playout point is calculated by averaging the collected playout points of all the active Sync Clients in the session. In the simulated case, the mean playout rate was slightly slower than the nominal playout rate (i.e., $\gamma_{master}=\gamma_{mean}<0$), resulting in an increasing playout delay that, depending on the session duration, could culminate in an overflow situation (unless additional techniques to avoid it are employed).

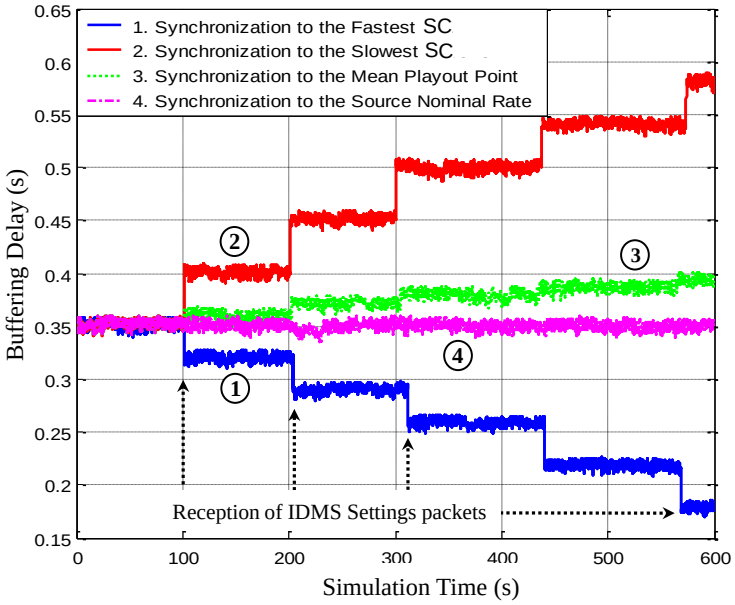


Figure 11.13. Buffering Delay evolution to Achieve IDMS for an ideal Sync Client when using SMS.

11.5 Evaluation of DCS

In this sub-section, the effectiveness of our IDMS solution when using DCS is tested. Figure 11.14 illustrates the same situation as in Figure 11.8 (i.e., group-based IDMS to the slowest Sync Client), but using DCS as the control scheme. Both graphs seem quite similar. However, we can observe in Figure 11.14 that the Sync Clients belonging to G1 (in which they were sparser than in G2) were not always simultaneously synchronized (e.g., at *200-th* and at *300-th* seconds). This is because when SC1 detected an out-of-sync situation (after gathering the IDMS reports from the other Sync Clients in that group), SC2 was still waiting for the IDMS report from SC1. This way, when SC1 sent an IDMS report, including its local playout point, SC2 did not detect that out-of-sync situation because SC1 had already begun (or even finished) its adjustment process.

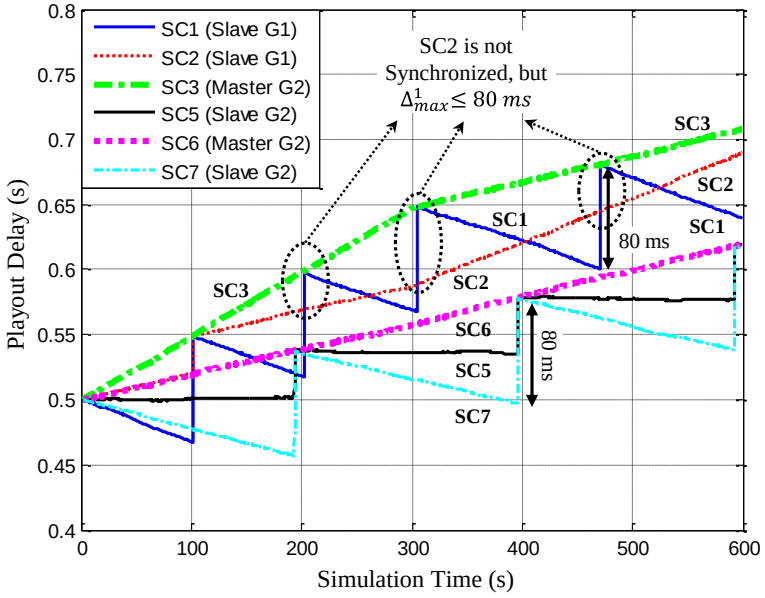


Figure 11.14. Playout Delay Evolution to Achieve IDMS: DCS using Synchronization Policy to the Slowest Sync Client.

The same effect for DCS can be appreciated in Figure 11.5, using the master selection policy to the mean playout point, and using AMP as reactive technique. It can also be seen that the three Sync Clients belonging to G1 did not always simultaneously enforce playout adjustments due to the previous described situation.

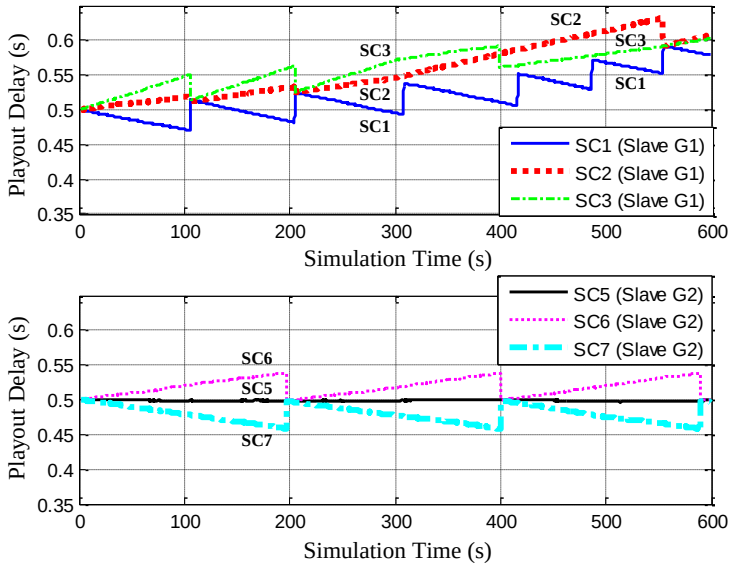


Figure 11.15. Playout Delay Evolution to Achieve IDMS: DCS using Synchronization Policy to the Mean Playout Point, using AMP.

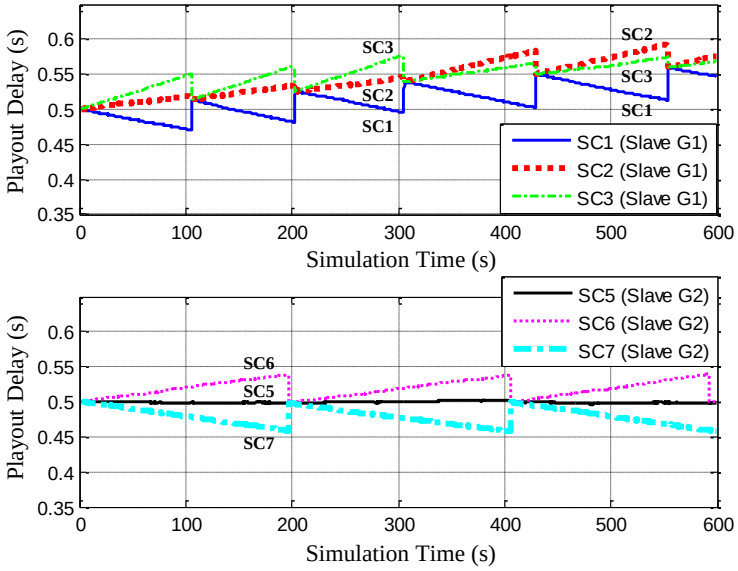


Figure 11.16. Playout Delay Evolution to Achieve IDMS: DCS using Synchronization Policy to the Mean Playout Point, using AMP + Coherence.

To overcome the above issue, the *coherence* adjustment technique (presented in Section 8.12) was enabled in the next simulation, the results of which are shown in Figure 11.16. In this case, all the Sync Clients were almost simultaneously synchronized to the same IDMS reference every time τ_{max} was exceeded in each group because, despite that in some cases they did not detect the out-of-sync situation, they did receive the IDMS reports notifying about the triggering of IDMS adjustments. This resulted in a more fine-grained synchronization and a clearly better performance in terms of coherence.

11.6 Comparison between SMS and DCS

This sub-section aims to provide some comparison results between SMS and DCS regarding interactivity and coherence.

11.6.1 Interactivity

Figure 11.10 (using SMS) and Figure 11.16 (using DCS) are almost indistinguishable. To clarify the differences among them, zoom views of the adjustments processes in both figures are shown in Figure 11.17 (for G1) and Figure 11.18 (for G2) for both SMS (left graphs) and DCS (right graphs). It can be seen that asynchrony situations were corrected earlier using DCS than using SMS, in both groups. Therefore, these graphics confirm that DCS outperforms SMS in terms of interactivity. This occurs because using DCS each Sync Client started the playout adjustments once it collected the IDMS reports from all the other Sync Clients in its own group. Using SMS, larger delays occur when exchanging the control information for IDMS. First, the Sync Manager must collect all the IDMS reports from the Sync Clients and, as shown in Figure 11.1 and in Table 11.1, there is a significant network delay between the Sync Clients and the Sync Manager (co-located with the Media Server). Second, the Sync Manager may not be able to send an immediate RTCP IDMS Settings packet after detecting an out-of-sync situation, since it has to adhere to the bounded RTCP timing rules (specified in RFC 3550 and explained in Section 7.5). Third, a copy of that IDMS Settings packet has to be received by all the Sync Clients to be able to enforce the required playout adjustments.

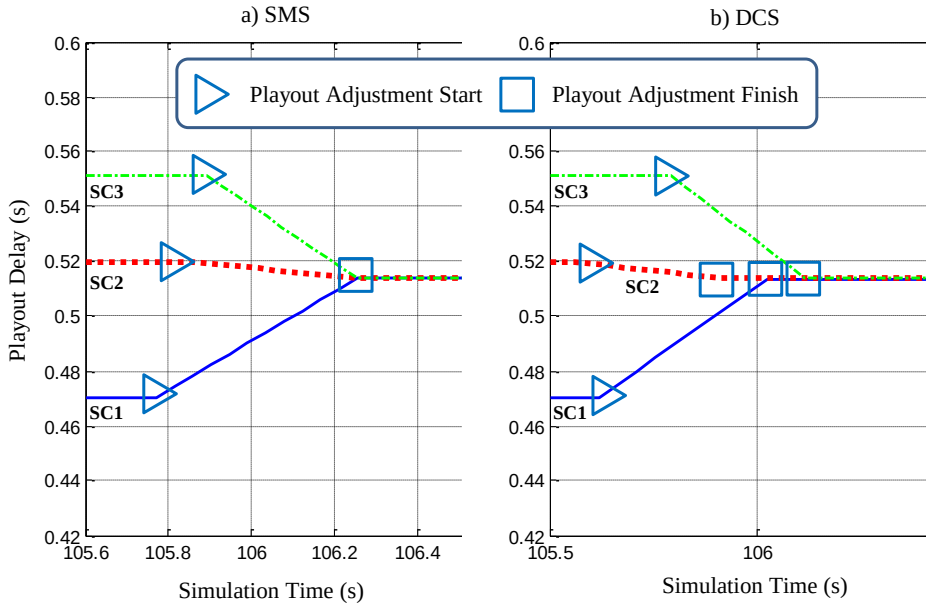


Figure 11.17. Zoom View of the Playout Adjustments (AMP) in Group 1.

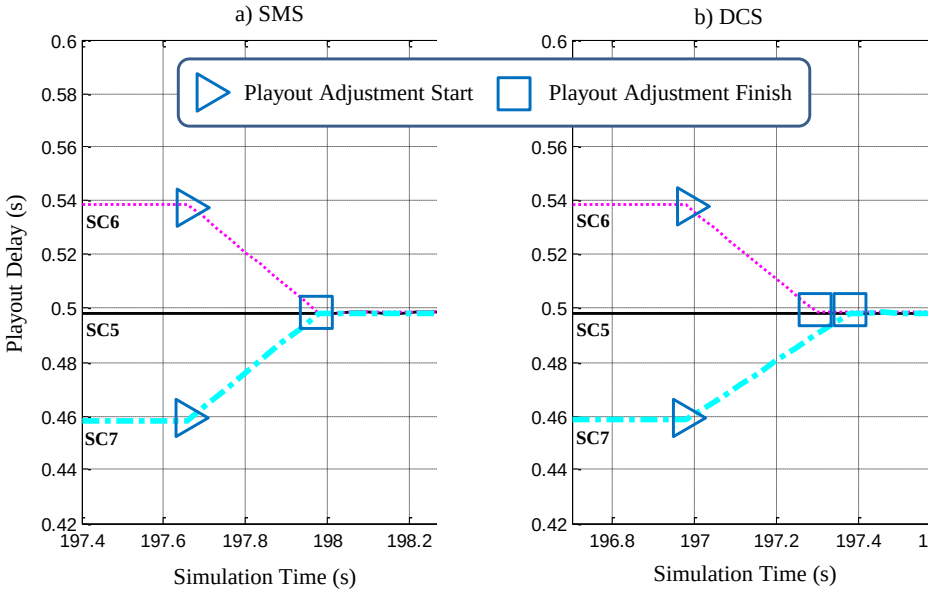


Figure 11.18. Zoom View of the Playout Adjustments (AMP) in Group 2.

Accordingly, the RTCP report interval, T_{RTCP}, plays a key role regarding the interactivity of our IDMS solution when using each one of the developed control schemes. As discussed in Section 7.5, T_{RTCP} is dynamically adjusted in each of the involved sync entities according to a local estimation of the population of the session and the available bandwidth, in order to avoid that the total amount of control traffic added by RTCP exceeds 5 % of the allocated RTP session bandwidth. In the simulated scenario, the Sync Manager was implemented within the single Media Server resources (i.e., $n_{senders}=1$) and the bandwidth for the RTP Session was configured to $BW_{session}=200$ *kbps*. Assuming an approximate value of the average size of all sent and received RTCP messages equal to 1000 bits (i.e., $avg(RTCP_{size})\approx1000$ bits, including the IDMS messages, plus UDP and IP layer headers), and according to formulas in Figure 7.3, a delay of up to 0.6 s could be accumulated between the instant at which an out-of-sync situation is detected by the Sync Manager and the instant at which it can transmit an IDMS Settings packet. This gives the maximum Sync Manager delay because of the bounded RTCP reporting rules. Such differences in terms of interactivity between SMS and DCS are relevant, especially for the IDMS use cases requiring stringent synchronization levels (discussed in Section 2.4). Even though the maximum playout asynchrony in each group may slightly increase during this additional Sync Manager delay, the issue here is that the out-of-sync situation will not be corrected during this time interval. For instance, as a result of 10 simulation runs, the maximum playout asynchrony in G2 during the session using SMS was 82.4 ms (mean value 39.4 ms) whilst the one using the DCS was 81.4 ms (mean value 38.8 ms).

It is important to emphasize that the magnitude of the Sync Manager delay would be significantly larger (up to 5 s or slightly superior) if either the minimum value for T_{RTCP} (from RFC 3550) or *trr-int* attribute (from RFC 4585) would have been considered in our simulation setup. Furthermore, if any of these parameters had been adopted, larger delays would be introduced when gathering the IDMS reports from the Sync Clients in each one of the developed schemes, thus also affecting to the interactivity of our IDMS solution. This is because T_{RTCP} in each Sync Client would be larger in such a case and, therefore, asynchrony situations would be detected later.

Finally, as can be appreciated in Figure 11.18, a significant advantage of using DCS for IDMS is that the time interval needed to exchange the IDMS messages can be significantly reduced if the Sync Clients are quite close to each other, as in G2 (see Figure 11.1).

11.6.2 Coherence

Regarding coherence, it can be also appreciated in Figure 11.17 and in Figure 11.18 that using SMS all the Sync Clients finished their adjustment processes almost simultaneously, because all of them were synchronized to the same target playout point included in the RTCP IDMS Settings packet. In such a case, the Sync Clients did not begin the playout adjustments at the same time because they received the

IDMS Settings at different instants (due to the variable network delays to the Media Server). Using DCS, however, each Sync Client started to adjust its playout process once it collected the overall playout status in its own group. Thus, the IDMS adjustment period did not begin/finish at the same time in all of them. It can be appreciated in the right graphs of both figures.

Additionally, Figure 11.19 shows that the playout rate was varied (using AMP) within tolerable limits during the session's lifetime in both SMS (Figure 11.19.a) and DCS (Figure 11.19.b), when using the master selection policy to the mean playout point. Moreover, the zoom views show that using SMS the playout adjustments were performed almost simultaneously in all Sync Clients (Figure 11.19.a), whilst this did not occur when using DCS (Figure 11.19.b). This also confirms that SMS is superior to DCS in terms of coherence (similar results were obtained for the other master selection policies).

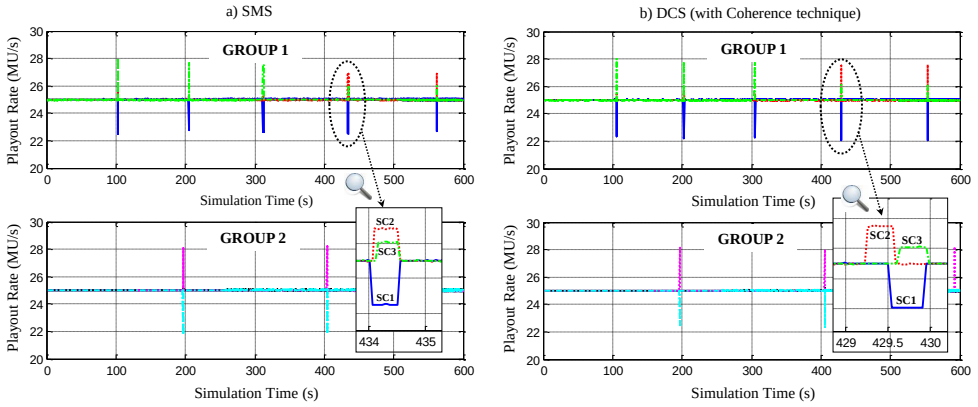


Figure 11.19. Playout Rate Variation: using Synchronization Policy to the Mean Playout Point, and using AMP.

Even though asynchrony situations and playout adjustments below the allowed threshold may be unnoticeable to users, coherence is an important feature to enable high accuracy after synchronizing, since all the Sync Clients will be almost simultaneously adjusted to the same IDMS reference. This can be especially important when multiple physically close-by Sync Clients need to be synchronized, because a single user may readily notice small delay differences (and subsequent playout adjustments), which may spoil the QoE. Therefore, the synchronization requirements become stricter and coherence is more relevant in such cases.

11.6.3 Playout Asynchrony Distribution

To complement the previous results, Figure 11.20 shows the distribution of the playout asynchrony (average results of 10 simulation runs) in G1 for the same master selection policy (synchronization to the mean playout point), when using SMS and DCS (enabling and disabling the coherence adjustment technique), and employing both aggressive and smooth playout adjustments.

Regarding the adjustment techniques, it can be appreciated that the Sync Clients were more accurately synchronized using AMP than using aggressive adjustments in both schemes, because the percentage of MUs played out with low asynchrony values was significantly higher, which corroborates our conclusions in Section 11.4 (summarized in Table 11.2). Conversely, large asynchrony values (near the threshold) were more probable using aggressive adjustments than using smooth adjustments.

Regarding the IDMS schemes, it can be appreciated that the percentage of MUs played out with low asynchrony ranges was significantly higher when SMS was employed. This is because in SMS all the Sync Clients were almost simultaneously synchronized to the same IDMS reference, because of the reception of RTCP IDMS Settings packets (high performance in terms of coherence), as shown in Figures 11.17, 11.18 and 11.19. It can also be seen that this percentage was significantly enhanced when enabling the coherence technique in DCS, as previously discussed. Conversely, the percentage of MUs that were played out with an asynchrony larger than the allowed threshold of 80 ms (i.e., out-of-sync MUs) was larger when using SMS than using DCS. This reflects the lower performance in terms of interactivity of SMS compared to DCS. As previously discussed, this percentage would have been significantly larger if any of either the minimum value for T_{RTCP} or *trr-int* had been adopted.

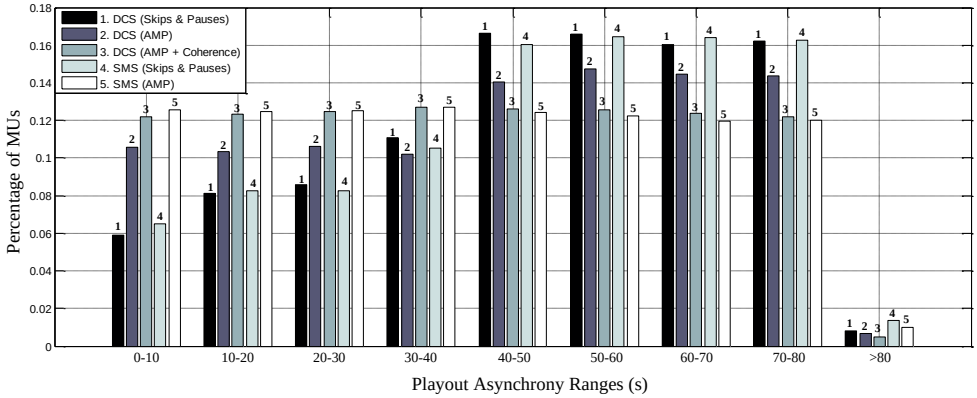


Figure 11.20. Playout Asynchrony Distribution: SMS vs DCS (Group 1).

Even though the percentages of the playout asynchrony distribution depend on several factors, such as the network scenario under test, the bandwidth availability, the number of active Sync Clients and their characteristics, etc., the values from Figure 11.20 are representative for illustrative purposes, corroborating the differences between the use of the different control schemes and adjustment techniques for IDMS.

11.7 Evaluation of EED RTCP Feedback for IDMS

This section aims to show the benefits of the proposed EED RTCP Feedback (Chapter 9) compared to the Regular RTCP Feedback when using SMS for IDMS.

11.7.1 Interactivity Comparison between Regular and EED RTCP Feedback

Figure 11.21 illustrates the playout delay evolution for three Sync Clients belonging to G1 to achieve IDMS when enabling the EED RTCP Feedback, by using the synchronization policy to the mean playout point. In such a case, every time an asynchrony exceeding τ_{max} (80 ms) was detected by the Sync Manager, it sent (multicast) an Early RTCP IDMS Settings packet to indicate the Sync Clients an target playout point to which they must synchronize (calculated by selecting the average playout timing of all of them as the IDMS reference). This figure seems equivalent to Figure 11.10.a, in which the same control scheme (SMS), the same master selection policy (synchronization to the mean playout point) and the same reactive adjustment technique (AMP) were employed. The difference between them is the RTCP Feedback mode in use: Regular RTCP (in Figure 11.10.a) and EED RTCP (in Figure 11.21). To clarify the differences among both cases, zoom views of the playout adjustment processes in each case are presented in Figure 11.22. It can be seen that the asynchrony situation was corrected later when using Regular RTCP Feedback (left graph) than when using EED RTCP Feedback (right graph). Therefore, this figure shows that EED RTCP Feedback outperforms Regular RTCP Feedback in terms of interactivity, mainly because of the minimization of the Sync Manager delay.

As discussed in Section 7.5, in the simulated scenario, a delay of up to 0.6 s could be accumulated between the instant at which an out-of-sync situation is detected by the Sync Manager and the instant at which it can transmit an IDMS Settings packet when using Regular RTCP Feedback. However, the Sync Manager delay can be minimized when using EED RTCP Feedback, because of the ability of the Sync Manager to send immediate RTCP packets as a response to the detection of events (asynchrony situations in this case). Besides, such delay differences between Regular RTCP and EED RTCP Feedback for IDMS would have been larger if any of the minimum value for T_{RTCP} (from RFC 3550) or *trr-int* attribute (from RFC 4585) would have been adopted, as also discussed when comparing the interactivity performance between SMS and DCS. Moreover, according to the RTCP timing rules, such delay differences would also be much larger if the Sync Manager functionality would have

been implemented as a part of an RTP receiver in a large-scale session (i.e., involving lots of Sync Clients), as can be inferred from the value of the RTCP report interval in Figure 11.23, or if there were multiple Media Servers in the session. Therefore, the proposed EED RTCP Feedback for IDMS would be even more beneficial in these two cases.

It is important to emphasize that these delay differences occur when using any of the policies for choosing the master reference for IDMS, although only the evaluation for the synchronization policy to the mean playout point has been included.

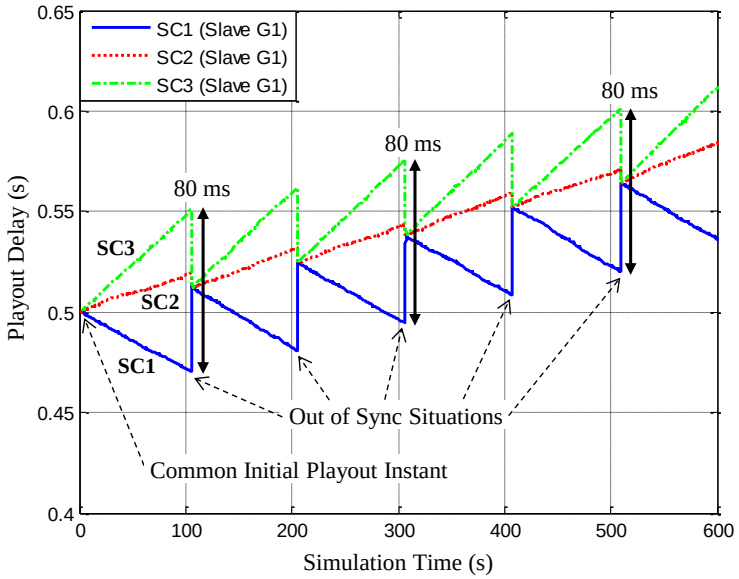


Figure 11.21. Playout Delay Evolution to Achieve IDMS: SMS using Synchronization Policy to the Mean Playout Point, using EED RTCP Feedback for IDMS.

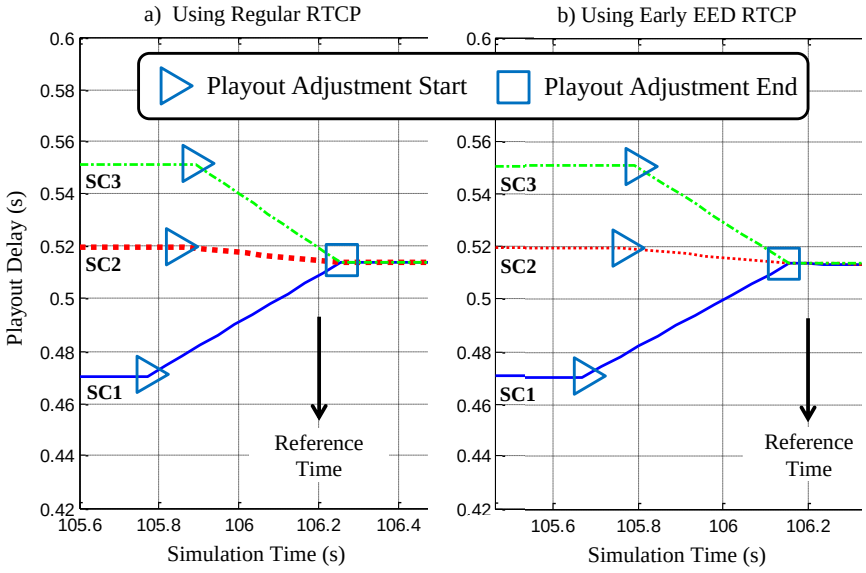


Figure 11.22. Zoom View of the Playout Adjustments to Achieve IDMS: Regular RTCP vs EED RTCP Feedback.

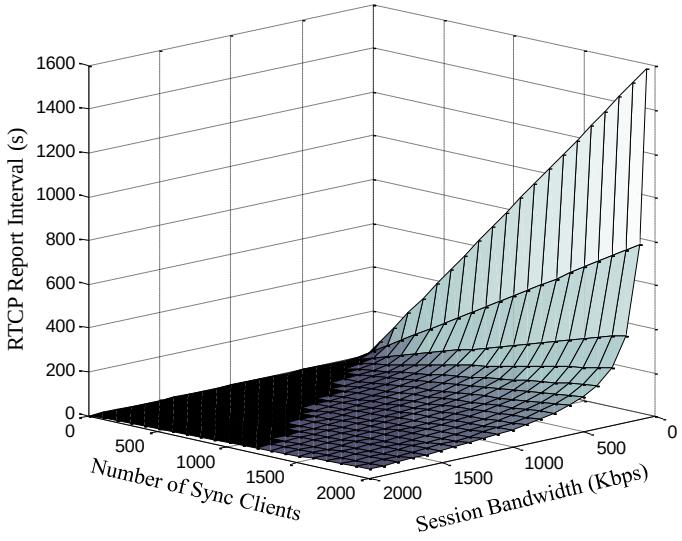


Figure 11.23. Adaptation of the RTCP Report Interval according to the Number of Sync Clients and to the Session Bandwidth.

To corroborate the benefits of using EED RTCP Feedback for IDMS, the fraction of MUs that were played out in all Sync Clients with an asynchrony larger than the allowed threshold was assessed, for different threshold values, when using both Regular and EED RTCP Feedback. Figure 11.24 shows the average results of 10 simulation runs (with different seeds for the random variables in each iteration). The following asynchrony threshold values were employed: sub-frame accuracy ($1/(2\cdot\theta)$=20 ms), frame accuracy ($1/\theta$=40 ms), 2 frames accuracy ($2/\theta$=80 ms) and 4 frames accuracy ($4/\theta$=160 ms).

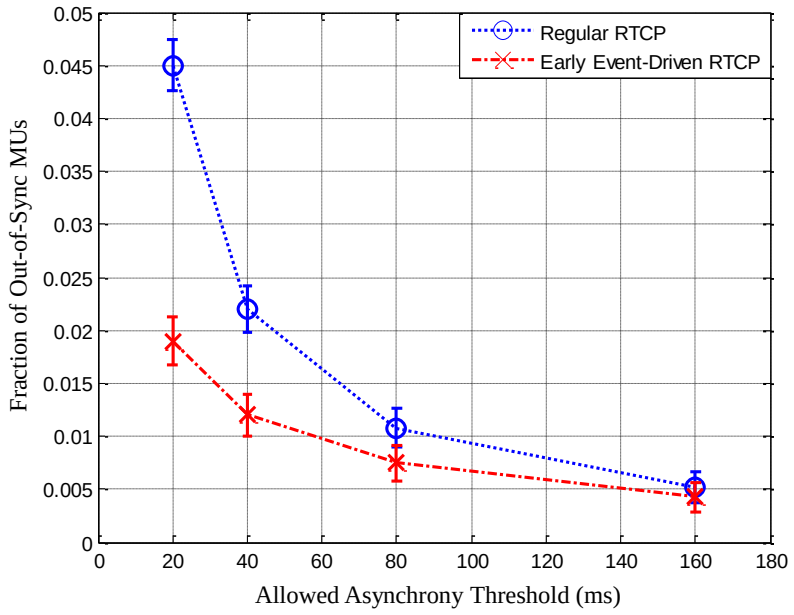


Figure 11.24. Interactivity Comparison between Regular and EED RTCP Feedback for IDMS.

The fraction of out-of-sync MUs when using EED RTCP Feedback (red graph) is not as dependent on the allowed threshold as when using Regular RTCP Feedback (blue graph), since a less steep slope can be observed in the graph. Indeed, we can see that despite the fact that the differences are not very high for the upper threshold (160 ms), they become relevant when the threshold is reduced. For example, the fraction of out-of-sync MUs is almost double for a frame accurate threshold (40 ms), whilst it is more than double for the lowest threshold, comparing Regular RTCP to EED RTCP Feedback. This corroborates the better performance in terms of interactivity, because of the earlier correction of out-of-sync situations, when using EED RTCP Feedback for IDMS. This is especially relevant for the IDMS use cases with stringent synchronization requirements. Also, it is important to mention that although the

percentages of out-of-sync MUs seem quite high, this fact does not mean that those asynchrony situations are annoying to human perception, because it is sufficient with setting an allowed threshold slightly lower than the noticeable asynchrony limits. For example, the maximum asynchrony value when using Regular RTCP Feedback for τ_{max}=80 ms in all simulations was 82.4 ms (higher than when using the EED RTCP Feedback), which confirms this assumption.

11.7.2 Fine Synchronization for Media-Related Events

Figure 11.25 illustrates the same situation as in Figure 11.21 when media-related events were triggered by the Media Server (through the Sync Manager) with a frequency of one event per 150 s (although they could have been dynamically or randomly triggered). In such a case, the Sync Manager sent IDMS Settings packets both as a response to the detection of out-of-sync situations (occurring in a non-deterministic way) and to the occurrence of media-related events (e.g., from application-dependent actions, such as post-advertisements, start of a football match, a penalty shot, a quiz in a TV show, user generated actions...), despite that the asynchrony at that moment was lower than the allowed threshold. Table 11.3 shows the synchronization granularity with which those events were presented in the involved Sync Clients when using both Regular RTCP and EED RTCP Feedback (mean value and standard deviation of 10 simulation runs). It can be seen that the asynchrony for media-related events can range from a perfect synchronization (i.e., no delay differences) to the allowed threshold (or a slightly superior value) when using Regular RTCP, because there is no provisioning for synchronizing dynamic media-related events. The asynchrony values from Table 11.3 for Regular RTCP Feedback can be checked through the graphical representation of the playout delays in Figure 11.21. In contrast, it can be seen that those media-related events were presented with highly accurate synchronization levels when using EED RTCP Feedback. The obtained synchronization granularity was not perfect (i.e., asynchrony equal to zero), as expected, mainly due to the configured playout rate deviations (Table 11.1) as well as to possible wall-clock synchronization mismatches.

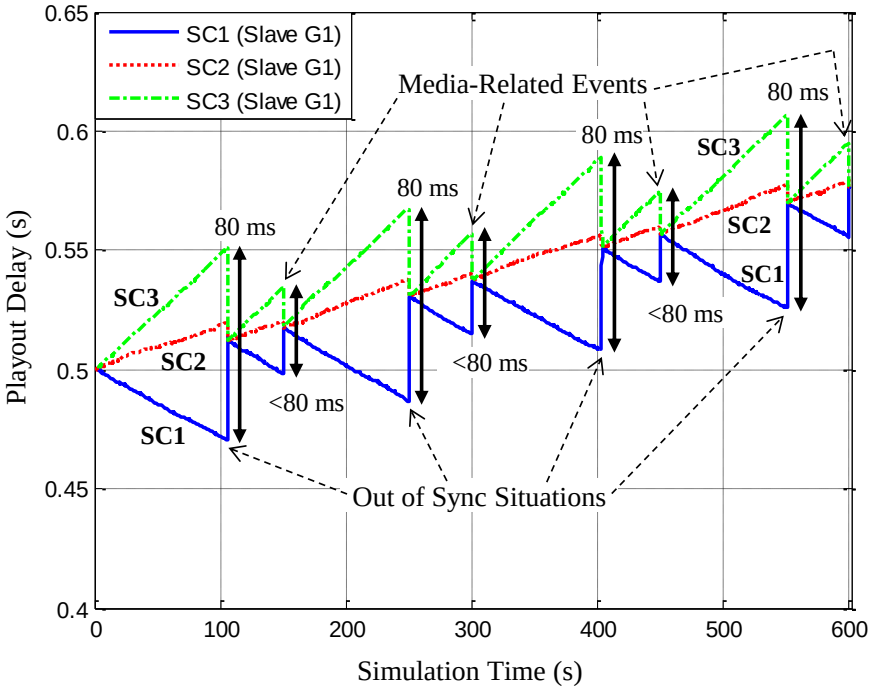


Figure 11.25. Playout Delay Evolution to Achieve IDMS: SMS using Synchronization Policy to the Mean Playout Point, using AMP and EED RTCP Feedback for IDMS (with Media-Related Events).

Table 11.3. Synchronization Granularity for Media-Related Events.

Event (s)	ASYNCHRONY (ms)	
	Regular RTCP	EED RTCP
E1 (150)	38.5 ± 0.627	0.080 ± 0.022
E2 (300)	77.0 ± 0.804	0.073 ± 0.017
E3 (450)	29.5 ± 0.832	0.066 ± 0.019
E4 (600)	70.7 ± 0.799	0.071 ± 0.018

11.7.3 Rapid Accommodation of Latecomers

Figure 11.26 illustrates the same situation as in Figure 11.21 when a latecomer (SC4) joined the session in progress at second 60, as an additional member of G1. At that moment, SC4 sent an RTCP-IDMS-REQ message, and started buffering the incoming RTP packets. Once the Sync Manager received the RTCP-IDMS-REQ

message, it immediately sent an Early IDMS Settings packet to allow SC4 to synchronize with the other Sync Clients in G1 as soon as possible. Once SC4 received the IDMS Settings packet, it scheduled its playout controller to be able to achieve IDMS (assuming SC4 was already able to start the consumption of the media stream, because of the adoption of some of the additional techniques discussed in Section 9.5). Depending on the target playout point included in the IDMS Settings packet, it could be possible that some buffered media data need to be discarded by the latecomer when starting its playout process. In the simulated cases, SC4 experienced a maximum IDMS latency of 2.1 s (Δt_1 = 0.8 s, RTT = 0.125 s, Δt_2 = 0.6 s, Δt_3 = 0.575 s, see Figure 9.4) when using Regular RTCP Feedback, whilst the one when using EED RTCP Feedback was decreased to 1.5 s (i.e., 2.1 – 0.6), mainly due to the fact that the Sync Manager delay (Δt_2 in Figure 9.4) was minimized. Therefore, as discussed earlier, the use of EED RTCP Feedback can also significantly contribute to decrease the zapping delays in media sessions requiring IDMS. Further research is needed to analyze the feasibility of reducing the other sources of delay (i.e., RTT, Δt_1 and Δt_3) when zapping in IDMS-enabled sessions. The first two (RTT and Δt_1) have been discussed in Section 9.5, whilst the third one (Δt_3) implies optimizing both the operation of the Sync Manager (IDMS target playout point calculation) and of the latecomer (buffering techniques and/or backward interpretation of the IDMS Settings packets).

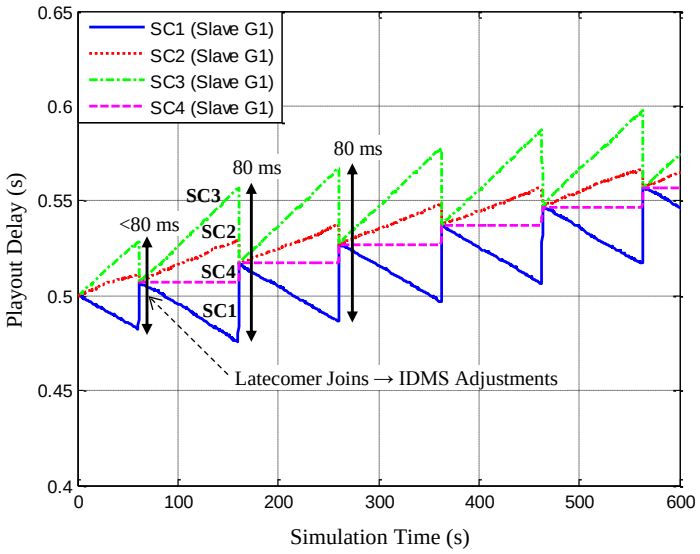


Figure 11.26. Rapid Accommodation of Latecomers (SC4) using EED RTCP Feedback.

11.8 Evaluation of M/S Scheme for IDMS

The goal of this sub-section is to test the satisfactory responsiveness of M/S Scheme for IDMS in our RTP/RTCP-based solution. When using M/S Scheme, the value of τ_{max} was set to 50 ms in all the slave Sync Clients with the aim of keeping the asynchrony bounds below 100 ms. This is because, as discussed in Section 5.3, using M/S Scheme, each slave Sync Client can only compute the asynchrony between its playout process and the one of the master Sync Client. So, the worst case would occur when the playout point of a slave Sync Client and the one of another slave Sync Client are extremely advanced and lagged, respectively, compared to the one of the master Sync Client. This differs from SMS and DCS, in which the playout asynchrony between any two Sync Clients can be known. For that reason, M/S Scheme has not been compared with SMS and DCS in terms of interactivity and coherence.

The playout delay evolution of three Sync Clients belonging to G1 (SC1, SC2 and SC3) and of three Sync Clients belonging to G2 (SC5, SC6 and SC7) to achieve IDMS when using M/S Scheme is shown in Figure 11.27 (using aggressive adjustments) and in Figure 11.28 (using smooth adjustments). As can be appreciated, SC2 and SC7 were the master Sync Clients in G1 and G2, respectively. It can be seen that the asynchrony between the playout states of the Sync Clients in each group progressively increased mainly due to the configured deviations in their local playout processes. Likewise, it can be seen in both graphs that the playout adjustments when using M/S Scheme were not simultaneously performed by all the slave Sync Clients. This is due to the fact that each one of the Sync Clients can only compute the asynchrony between its local playout point and the one of the master. That confirms the lower performance in terms of *coherence* compared to SMS.

When using aggressive adjustments (Figure 11.27), it can be seen that every time lagged Sync Clients in G1 (SC3) detected an asynchrony between their local playout point and that of the master Sync Client in that group (SC2) exceeding the allowed threshold (τ_{max}=50 ms), they adjusted their playout timing by skipping just one MU[45] to achieve IDMS (see zoom view in the upper graph). Therefore, there was a residual asynchrony of around 10 ms (i.e., τ_{max}-$1/\theta \approx$ 50-40 ms) that was not corrected. Conversely, advanced Sync Clients (SC1) had to pause their playout process every time τ_{max} was crossed in order to synchronize with the master in G1 (SC2). In such a case, advanced Sync Clients in G1 did acquire a more fine-grained synchronization than lagged ones because they paused specific MUs the exact time corresponding to the value of the detected asynchrony (Δ^{k}_{max}) in order to minimize it. In G2 (lower graph in Figure 11.27), the master Sync Client (SC7) was the most advanced (i.e., the one with the lowest playout delay: $d_{master}=d_7 \leq d_6 \leq d_5$). Therefore, slave Sync

[45] It is assumed that only entire MUs can be skipped, each one with a duration of $1/\theta=1/25$ = 40 ms/MU.

Clients (SC5 and SC6) had to skip MUs to reduce the detected asynchrony every time τ_{max} was exceeded.

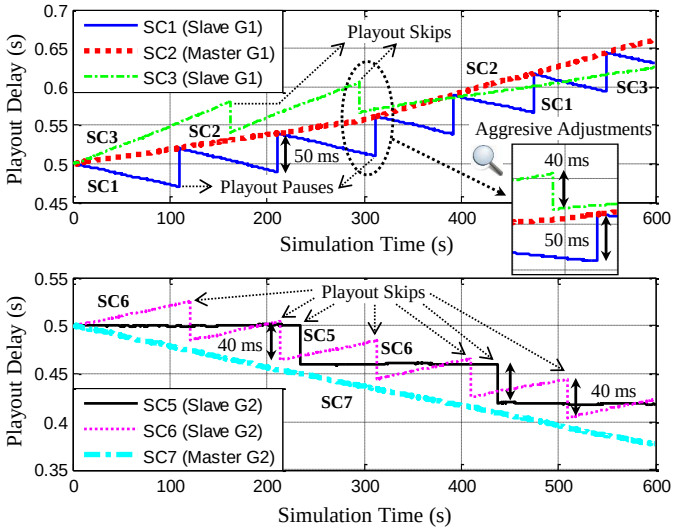


Figure 11.27. Playout Delay Evolution to Achieve: M/S Scheme, using Aggressive Adjustments.

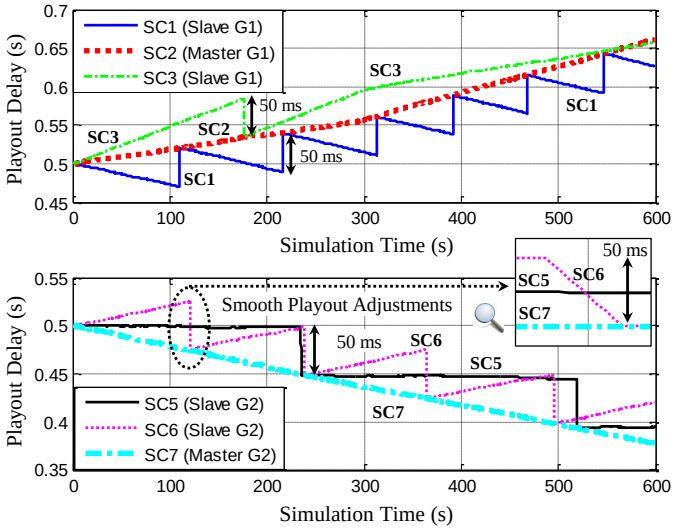


Figure 11.28. Playout Delay Evolution to Achieve IDMS: M/S Scheme, using AMP.

When using AMP (Figure 11.28), the above long-term playout discontinuities were avoided for both lagged (skips) and advanced (pauses) slave Sync Clients. In addition, lagged slave Sync Clients were more fine-grained synchronized than using aggressive adjustments because they smoothly increased their playout rate to minimize the detected asynchrony (see zoom view in the lower graph of Figure 11.28).

In the two previous figures, it can also be seen in the upper and lower graphs, a significant increase and decrease, respectively, of the playout delays in all the Sync Clients as the session advanced in time, which caused an inherent progressive filling or emptying, respectively, of their playout buffer occupancy. This confirms the assumptions in Section 8.8. Thus, if the master Sync Client is significantly advanced or lagged, the playout buffers of all Sync Clients may suffer overflow or underflow, respectively, if the media session had a long duration. Therefore, the use of M/S Scheme for IDMS would require the use of additional adaptive techniques, such as buffer fullness monitoring and control, to avoid such situations.

Figure 11.29 corroborates that the playout adjustments in Figure 11.28 were performed by all the slave Sync Clients within perceptually tolerable ranges, with a variation of the playout factor always within allowable bounds (i.e., $|\varphi^{k}_{max}| \leq 0.25$).

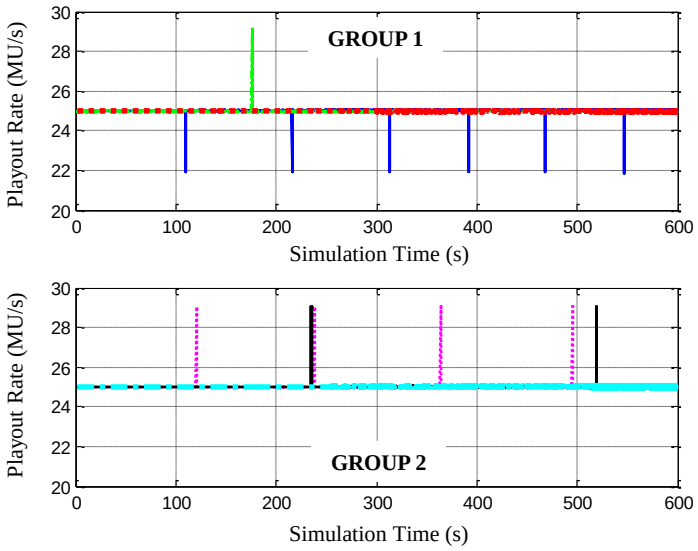


Figure 11.29. Playout Rate Adjustments: M/S Scheme, using AMP.

To conclude the evaluation of M/S Scheme for IDMS, the same scenario when a new ideal Sync Client with a perfect playout rate (without deviations) joined the multicast session, as an additional member of G1, was tested. The playout controller

of that Sync Client was able to play out the incoming MUs with the same rate as they were generated by the Media Server. In that Sync Client, the buffering delay for the incoming MUs was measured when each one of the Sync Clients in G1 were selected as the master. It can be seen in Figure 11.30 that the buffering delay in that Sync Client was kept quite uniform during the multimedia session (the appreciated fluctuation is due to the jitter delays) until the reception of a new IDMS report from the master Sync Client. At that moment, that Sync Client had to adjust its playout timing in order to correct the estimated playout time discrepancy, according to the reported playout timing by the master Sync Client in its IDMS report. As expected, when SC2 and SC3 (slow Sync Clients) were selected as the IDMS master references, the buffering delay for the incoming MUs increased after the reception of each IDMS report. This resulted in an inherent filling of its playout buffer that could even overflow in long multimedia sessions. On the contrary, when SC1 (the fastest Sync Client) was selected as the master, the buffering delay in this Sync Client decreased after the reception of each IDMS report from SC1. As a consequence, the buffer fullness level of that Sync Client was progressive decreasing. Finally, when SC4 was selected as the master reference for IDMS, its playout delay presented a smooth evolution, because its ability to play out the media with a *'perfect'* timing. Consequently, the buffer fullness level was moderately stable during the multimedia session.

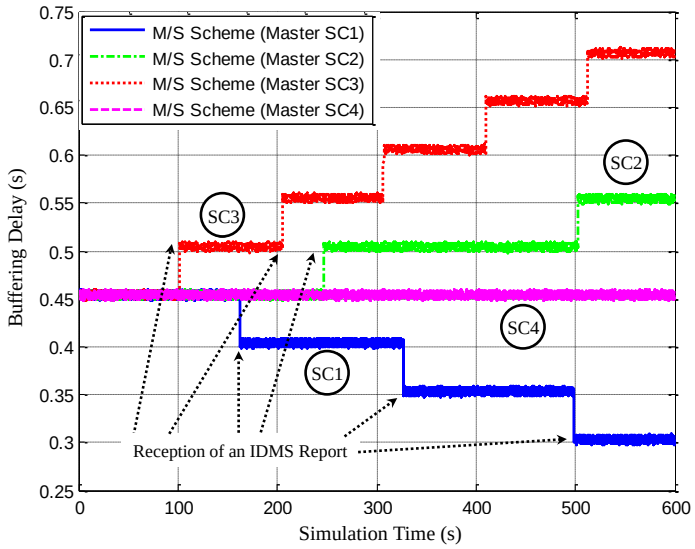


Figure 11.30. Buffer Delay Evolution to Achieve IDMS for an ideal Sync Client when using M/S Scheme.

11.9 Traffic Overhead

As discussed in Section 7.5, in standard compliant RTP/RTCP streaming scenarios, the total amount of RTCP traffic must be bounded to 5 % of the allocated RTP session bandwidth. The newly defined RTCP messages for IDMS only constitute a small subset of all existing RTCP packets and reports, and are typically sent in compound RTCP packets. Concretely, the RTCP compound packets are typically larger than 120 bytes, whereas the size of an IDMS report is 40 bytes and the size of an IDMS Settings packet is 36 bytes. Accordingly, the traffic overhead added by our IDMS solution will be always significantly lower than this percentage when using each one of the control schemes, independently on the number of active Sync Clients in the session (and on the number of involved groups), because the RTCP report interval will be dynamically adjusted to adhere to the RTCP traffic bounds (RFC 3550).

In our simulated scenario, when using SMS with Regular RTCP Feedback, the total number of IDMS reports sent (multicast) by each Sync Client during the 10-minutes session was around 2% of the total RTP data packets sent by the Media Server (around 15000 RTP packets), whereas the total number of RTCP packets sent by the Sync Manager (co-located with the Media Server) was around 8 % of the total number of RTP packets, with slight variations in each simulation (depending on the initial seed). From the total number of RTCP packets sent by the Sync Manager, the number of IDMS Settings packets depends on the detected asynchrony between the playout states of the Sync Clients in each group. As an example, when the synchronization policy to the mean playout point was employed (Figure 11.10), 5 packets were sent to G1 and only 3 packets to G2 (because the playout deviations in that group were minor). The percentage of sent IDMS reports are very similar to the percentage of sent RTCP RR EXT packets in [Bor08]. However, the number of sent IDMS Settings packets is lower than the number of sent RTCP APP ACT packets in [Bor08], probably because in that case each Sync Client had to play out three different media types and the Sync Clients could become overloaded at some instants.

When using DCS, there were no significant differences regarding the number of IDMS reports sent by the Sync Clients. However, only one RTCP IDMS Settings packet was sent per each stage in which the session is divided (to indicate its starting instant), since each Sync Client locally calculates and performs the required adjustments according to the incoming IDMS reports.

When using M/S Scheme, only the master Sync Clients sends IDMS reports, so the number of IDMS reports sent by this Sync Clients was slightly superior than using the other IDMS schemes. This is because the average size of all sent and received RTCP packets was slightly inferior (as no IDMS reports were sent by slave Sync Clients).

Similarly, the same number of RTCP packets were sent by the Sync Manager when using SMS with RTCP Feedback and with EED RTCP Feedback, always adhering to the allowed RTCP traffic bounds. When using EED RTCP Feedback, Early IDMS Settings packets were sent in specific situations (e.g., initial playout instant, media-related events, out-of-sync situations, zapping…). However, the next Regular RTCP transmission time was skipped (as explained in Section 9.3). Therefore, the total number of sent RTCP packets by the Sync Manager did not differ.

The newly defined RTCP messages for IDMS (standardized in RFC 7272) are a bit larger than the proposed RTCP extensions in [Bor08] and in [Bor09c]. However, this increase of size only minimally affects the frequency of RTCP reporting, according to the timing rules in RFC 3550. Therefore, the traffic overhead for IDMS is very low, because we have not defined a new proprietary protocol, but we have taken advantage of the RTCP extension capabilities for designing our own adaptive and standardized IDMS solution.

11.10 Computational Load

Regarding the computational load, when using SMS, the Sync Manager must process all the IDMS reports from all the Sync Clients in the session (N_T), although the IDMS control is performed separately for each of the involved groups. Contrarily, the distributed Sync Clients have to process a low number of IDMS Settings packets from the Sync Manager as a response to specific situations (e.g., out-of sync situations, zapping, media-related events…). When using DCS, each Sync Client must process a number of IDMS reports given by $(N_G\text{-}1)/N_T$ per each RTCP report interval, where N_G is the number of Sync Clients belonging to a specific group G, and N_T is the total number of Sync Clients in the session. Using M/S Scheme, each Sync Client must only process one IDMS report (from the master Sync Client) per each RTCP report interval.

Finally, it is important to point out that the detection of out-of-sync situations by the Sync Clients, either locally computed when using DCS and M/S Scheme or notified through an IDMS Settings packet when using SMS, will result in the calculation of the necessary playout adjustments to achieve IDMS. This will depend on many aspects of the targeted scenario, such as the number of Sync Clients, their characteristics, the allowed threshold, the master selection policy in use, and the adjustment technique in use. As an example, Table 1.1 summarizes the number and percentage of MUs that were affected by the IDMS adjustment processes when using SMS.

11.11 Summary

The evaluation results in this Chapter have proved the satisfactory responsiveness of our IDMS solution, as well as its consistent behavior, when using each one of the deployed architectural schemes, master selection policies, control algorithms and adjustment techniques.

First, the implications of adopting the four discussed master selection policies and the two types of reactive playout adjustment techniques, when using SMS, have been analyzed. On the one hand, it has been shown the impact of the selection of the different master selection policies in terms of the synchronization effectiveness, delays and playout buffer occupancy. On the other hand, it has been proved the better convenience of using AMP than using aggressive playout adjustment techniques, for each master selection policy and control scheme in use.

Second, the effectiveness of DCS for IDMS has been shown, and the effects of applying or not the *coherence* technique have been verified. Likewise, the differences between SMS and DCS, in terms of coherence and interactivity, have been shown, corroborating the assumptions in Section 5.3.

Third, the simulation results have provided evidence of the ability of the designed EED RTCP Feedback to achieve faster reaction to specific situations (e.g., out-of-sync situations or channel change delays) in IDMS-enabled sessions, as well as a finer granularity for syncing dynamic application-to-media events in all Sync Clients, compared to using Regular RTCP Feedback, while still adhering to the RTCP traffic bounds.

Fourth, the effectiveness of M/S Scheme has also been shown.

In general, the obtained results have shown the ability of the designed RTP/RTCP-based IDMS solution, for each control scheme and master selection policy in use, to keep the asynchrony between the playout states in different groups of Sync Clients within acceptable limits, while minimizing annoying long-term playout discontinuities (skips/pauses), and hardly increasing the network and computational load.

Chapter 12

CONCLUSIONS AND FUTURE WORK

12.1 Conclusions

Along this dissertation, the summary and conclusions of each individual chapter have been provided. This Chapter includes the conclusions of this PhD thesis from a more general perspective.

In this PhD thesis, a thorough review to the multimedia synchronization area has been provided, paying special attention to IDMS. Likewise, the emerging distributed media consumption paradigm has been analyzed, with the goal of identifying the associated challenges, relevant use cases and existing IDMS solutions. Up to 20 use cases in which IDMS is necessary or beneficial have been compiled, and (qualitatively) classified according to their synchronization requirements. Moreover, it has also been shown that delay differences in current delivery networks are significantly larger than tolerable limits in the compiled use cases, thus revealing the need for IDMS.

Based on the review to the state-of-the-art and on the analysis of the emerging use cases, the necessary components (e.g., delivery and control protocols, architectural schemes, control and signaling mechanisms, adjustment techniques…), potential alternatives for such components, and their interaction to efficiently provide IDMS, have been identified.

Similarly, key requirements for IDMS have been derived. These requirements have been the basis to accomplish the *main goal* of this PhD thesis, which was the design, development and evaluation of an inter-operable, adaptive and accurate IDMS solution.

The core component of an IDMS solution is the media delivery and control protocols in use. As discussed, RTP/RTCP standard protocols (RFC 3550)

inherently meet several of the derived requirements, and their extension capabilities allow fitting the remaining ones. Therefore, such protocols were selected as the main candidates for being extended for IDMS, rather than specifying (proprietary) ad-hoc protocols. Concretely, two new RTCP messages have been specified to enable the concurrent and independent synchronization of the media playout of multiple groups of users in a shared media session. The use of RTP/RTCP for IDMS provides many advantages, such as widespread support, availability of an adaptive feedback channel, inherent rate adaptive mechanisms, as well as support for multiple media types and use cases (e.g., Social TV, multi-party conferencing, CoD, distributed audio systems…). Moreover, it allows synchronizing media streams at the packet level, while providing accurate (end-to-end) synchronization. Such RTCP extensions for IDMS have been standardized within the IETF, in RFC 7272.

In addition, novel EED RTCP reporting rules and feedback messages have been designed to improve the performance of the proposed IDMS solution in terms of dynamism, flexibility, interactivity and accuracy. It has been discussed and proved the benefits and potential impact of such mechanisms in current DTV deployments, especially for enabling dynamic content-based synchronization adjustments and for helping to reduce channel change delays. These RTCP extensions for IDMS have been published in an IETF internet draft [Mon15], which will be presented at the next 92th IETF meeting to study the convenience of their standardization.

The availability of standard mechanisms for IDMS will assure inter-operability between implementations, even when third-party infrastructure and communication devices are involved. Standardization eases the integration of the developed streaming capabilities as part of the available media frameworks (e.g., operating systems, media players...), which will also help to promote deployment of innovative (synchronization-sensitive) media services.

A second key component of the designed IDMS solution is the adoption of the proper architectural schemes to exchange the necessary information for IDMS. The existing control schemes for IDMS have been exhaustively compared, in a qualitative manner, based on many key aspects. Due to their different strengths and weaknesses, our IDMS solution has been adapted to adopt both centralized (SMS and M/S Scheme) and distributed schemes (DCS). This will allow to efficiently deploy the IDMS solution in a wide range of scenarios, according to the targeted use cases (e.g., Social TV, e-learning, audio beamforming…), the specific features (e.g., interactivity, scalability, accuracy, coherence…), and the characteristics and available resources of the networked environment (e.g., multicast support, delays, bandwidth…). Moreover, specific control mechanisms have been devised to enhance the performance of such control schemes for IDMS, such as fault tolerance methods, and algorithms to enhance the performance in terms of scalability (group-based synchronization control in all the considered control schemes), coherence (for DCS), and interactivity (for SMS, by adopting EED RTCP Feedback).

A third important component of the IDMS solution is the supported control mechanisms, such as dynamic strategies for selecting a reference timing to synchronize with, asynchrony monitoring and calculation methods, control timers, etc. For instance, this PhD thesis has explored the feasibility and suitability and dynamic strategies for selecting a master reference to synchronize with, and the impact of adopting such strategies on the synchronization effectiveness, delays, playout buffer occupancy levels, etc.

Moreover, a novel Adaptive Media Playout (AMP) technique for IDMS has been proposed, which is targeted at smoothly adjusting the media playout rate, within perceptually tolerable limits, every time asynchrony situations need to be corrected. This way, long-term and annoying playout discontinuities, as a result of the required synchronization adjustments, can be avoided.

The proposed IDMS solution has been tested through simulation. The obtained results have shown the ability of the designed RTP/RTCP-based IDMS solution, for each control scheme and master selection policy in use, to keep the asynchrony between the playout states in different groups of Sync Clients within acceptable limits, while minimizing annoying long-term playout discontinuities (skips/pauses), and hardly increasing the network and computational load.

Each one of the initially targeted goals (enumerated in Section 1.2) have been successfully achieved, and either conclusive answers to or valuable insights about the formulated research questions have been provided. Therefore, the research work has been completed.

The proposed technological components, the discussions about their suitability and provided (comparison) results can be a valuable source of information for researchers and developers interested in synchronization-sensitive distributed media systems.

The research in this PhD thesis has led to many publications (listed in Appendix A) in high quality journals (e.g., IEEE Communications Magazine, Multimedia Systems, Computer Networks, IEEE Communications Letters…) and conferences (e.g., IEEE LCN 2011 and 2013, ACM MM 2013, ACM TVX 2014…). Likewise, the interest of standardization bodies (ETSI TISPAN and IETF), whose members are recognized experts from both academia and industry around the world, for adopting our RTP/RTCP-based technology for IDMS reflects the impact of the research carried out within the context of this PhD thesis. In addition, it is important to mention that local and international companies have shown interest in the research carried out within the context of this PhD thesis, which hopefully will be translated into knowledge transfer from academia to industry. This is also a proof of the relevance and timeliness of the covered research topic. Accordingly, we believe this PhD thesis provides strong contributions to this research area.

12.2 Future Work

It is beyond doubt that multimedia synchronization, and particularly IDMS, will remain a live research area for the time to come, as emerging synchronization challenges continuously arise in the current multi-protocol, multi-sensory and multi-device delivery ecosystem.

Further research on distributed media synchronization needs to be targeted on several topics, such as optimizing the involved technological aspects (e.g., delivery platforms and protocols, clock synchronization, adjustment and control techniques, implementations…), on integrating novel media types (e.g., multi-sensory and high resolution media data), and on supporting emerging patterns in media consumption (e.g., hybrid broadband and broadcast communication, personalization features, multi-screen settings…).

We believe in the relevance of IDMS in current and future media delivery deployments. Accordingly, we have in mind several plans for future work on IDMS, which are briefly discussed in this Section.

12.2.1 Advanced IDMS use cases in RTP streaming

This sub-section provides a discussion about the need for standard compliant solutions in relevant and advanced IDMS use cases using RTP streaming. These solutions are needed to meet further synchronization demands, which are not (properly) dealt with the current IDMS solution designed in this PhD thesis.

12.2.1.1 IDMS for Different Streams

In shared media experiences, it could be possible that different users need to be synchronized (by means of IDMS), but are receiving different streams. As shown in Figure 2.4, the designed IDMS solution in this PhD thesis is mainly applicable when the different users are consuming the same media stream, either in a unicast or multicast way. Accordingly, further extensions for IDMS will be needed to accomplish these requirements. These different media streams can contain different formats of the same media content (e.g., HD and SD streams), different views of the same scene (e.g., from different cameras in a stadium or in a circuit race) or even different media types (e.g., audio and video). Likewise, such media streams can be generated by either the same or different servers. Another interesting functionality is the support of (in-network) trans-coding by intermediate servers to account for variable network conditions in specific (sub-)domains. Besides, a similar issue is the concurrent synchronization of different unicast CoD streams, including the same media content, independently requested by different users.

The challenges for providing IDMS in such cases are to correlate the different RTP streams (in terms of their relationship with a specific multimedia content and in terms of timelines) and to simultaneously compensate the delay differences for each one of them when reaching the involved destinations.

ETSI TISPAN proposal for IDMS [ETSI TS 183 063] contains mechanisms for synchronizing the same IPTV content in different quality formats, including trans-coding and re-origination of RTP streams (with the consequent changes in RTP headers including metadata for synchronization purposes). We believe in the convenience in continuing, improving and completing this initial work in order to provide full-fledged standardized solutions for all the previous use cases. Consequently, we are actively involved, as co-authors, in a new standardization work within the IETF to devise the proper solutions for these advanced IDMS scenarios [Sto14].

12.2.1.2 Scalable IDMS Solution

Some IDMS use cases, such as IPTV, Internet Radio or MOGs, can involve a lot of users. In such situations, according to the RTCP timing rules from RFC 3550, the RTCP report interval in each Sync Client will increase as the overall population in the session grows (see Figure 11.23). This will lead to a not enough frequent exchange of RTCP messages, providing outdated and unusable statistics. This is unacceptable for guaranteeing IDMS in such large-scale scenarios.

In addition, the traditional model of many-to-many (i.e., multicast) group communication may be either not available, or not desired (or not optimal) in the above scenarios. Accordingly, the Single-Source Multicast (SSM) with unicast feedback model was proposed in RFC 5760 [Ott10a]. Using this communication model, apart from the Media Sender and receivers, additional entities are involved in the media distribution process, such as Feedback Targets (FTs) and the Distribution Source (DS). This is illustrated in Figure 12.1.

We foresee the following RTCP extensions to enable the deployment of a really scalable IDMS solution in such large-scale SSM scenarios. First, timing correlation information and coordination functions between the Media Sender, the DS and the Sync Manager need to be specified (if these entities are not co-located). This will imply adopting a hierarchical architecture for IDMS, involving independent RTCP domains and additional active entities in the IDMS control processes (such as FTs and the DS). Second, additional (unicast) RTCP feedback collection and aggregation mechanisms for IDMS will be needed when FTs are present in a SSM scenario. This will involve the definition of new sub-report blocks (RFC 5760) for carrying out aggregated statistics about IDMS timing. This is targeted to enable a more efficient usage of the RTCP channel for IDMS, thus offering the required scalability for synchronizing all the Sync Clients in such large-scale sessions. In addition, it could be possible that not all the users require IDMS, but only some of them want to be synchronized together. This is for example the case when a group of friends want to share a TV experience together, but are not interested in sharing such experience with the other users outside their group also watching the same TV event. In such a case, the transmission of multicast IDMS setting instructions would lead to a situation where all Sync Clients would receive a multitude of different

setting instructions. They would have to find their own instructions based on their SyncGroupId, which is possible. However, with a large number of groups and of users, this would be highly inefficient. That is why additional mechanisms for notifying IDMS settings instructions to the Sync Clients belonging to small groups are required.

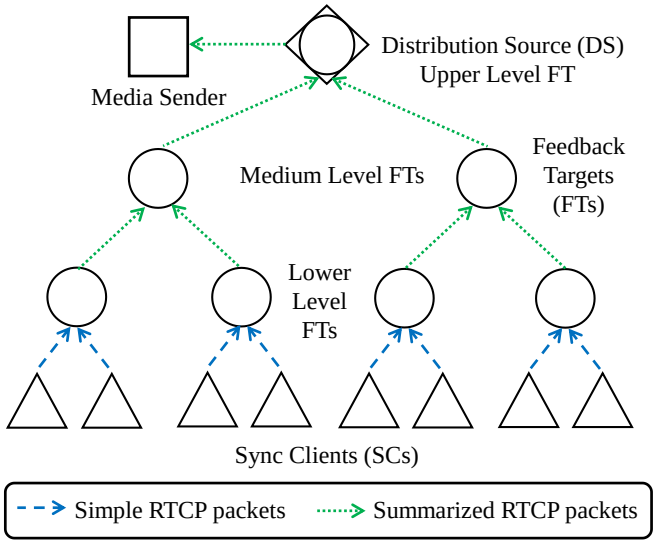


Figure 12.1. SSM Hierarchical Architecture.

12.2.2 IDMS in Web-Based Technologies

Due to the potential and promising future of WebRTC, research will also be needed to adapt and/or adopt the current RTP/RTCP media synchronization mechanisms to support the envisioned (multi-party) WebRTC use cases [Hol14]. This would allow providing synchronized multi-party media services via the web browsers, without needing to install (third-party) media servers and players.

Moreover, media synchronization issues are not only restricted to RTP/RTCP, but other forms of streaming technologies, such as the different variants of HTTP streaming and segmented video delivery are gaining momentum, and also require synchronization features. Therefore, research is needed to explore and/or to extend their distributed media synchronization capabilities, as for RTP/RTCP. The work in [Rai14] to provide IDMS in DASH technology is an initial step in that direction.

12.2.3 Context-aware IDMS

The design of the IDMS solution in this PhD thesis has been mainly focused on technological issues (e.g., protocols, architectural schemes, control mechanisms, adjustment techniques…). Acknowledging the relevancy and necessity of optimizing the technological aspects, coordination with social and psychological factors, commonly referred to as context [Tim14], is also required to maximize the overall perceived QoE in IDMS-enabled systems.

The challenges are not only to adapt the communication system, and therefore the synchronization control, according to changes at the network layer (e.g., bandwidth availability, delay, jitter…) and at the application layer (e.g., encoding settings, semantic information, type of media content and their importance…), but also according to the human, social and context layers [Ces14].

12.2.4 QoE Perception Tests

We plan to use our developed prototypes in real media frameworks to subjectively assess the benefits on the QoE provided by IDMS. To accomplish this, subjective testing in different scenarios and uses cases need to be conducted, complementing the preliminary studies and findings in [Gee11]. The user perception tests will be relevant to help answering the following research questions: i) which asynchrony levels are noticeable and annoying to users; ii) if users feel more together when IDMS is provided than when it is not (i.e., the impact on *"networked togetherness"*); iii) which architectural schemes and adjustment techniques are best suited for IDMS; iv) which interaction channel provides higher level of immersion and increases the feeling of *"togetherness"*; and v) the synchronization accuracy that can be achieved in real networked scenarios, etc.

These subjective tests need to involve large groups of users (existing studies have focused on small groups of users) in natural domestic environments, rather than in artificial lab settings (as in most related studies).

A set of questionnaires will be designed and multiple interviews will be conducted to investigate the influence of several aspects on the perceived QoE. Statistical analysis will also need to be performed to confirm the validity of the experimentation.

However, research in this context is not limited to perform the evaluations, analyze the results and derive the consequent conclusions. QoE evaluation for shared media services is still in its infancy [Tim14]. Currently, standards and methodologies for assessing the QoE in shared media experiences are non-existent. Moreover, these evaluation methodologies must not only be limited to technological issues, but must also take into account other relevant aspects, such as the media and content types, and the context, as discussed in the previous sub-section.

Accordingly, further research is also need to find out the most proper evaluation methodologies and metrics, as well as the specific factors to take into account.

12.2.5 Optimized AMP Techniques

Further research on AMP for IDMS will also be addressed. Even though the designed AMP technique (see Section 8.11) enables the achievement of an overall synchronization status, while keeping the playout rate variation within allowable limits, two additional goals will be pursued. The first one is to find out the best curve for minimizing abrupt changes in the media playout rate (i.e., slowly accelerate or decelerate), while keeping the playout rate variation within allowable bounds and not enlarging too much the synchronization adjustment period. The second one is to meet a trade-off between a proper playout buffer occupancy in all Sync Clients and the overall IDMS control. User perception tests will be conducted to analyze the benefits on the QoE provided by these optimized AMP techniques.

12.2.6 Standardization

We also plan to continue with our efforts for standardizing IDMS technology as done in last years.

REFERENCES

[Ade09] O.A. Ademoye, G. Ghinea, "Synchronization of Olfaction-Enhanced Multimedia", IEEE Transactions on Multimedia, vol.11, no.3, pp.561-565, April 2009.

[Aga11] P. Agarwal, R. Rivas, W. Wu, A. Arefin, Z. Huang, K. Nahrstedt, " SAS kernel: streaming as a service kernel for correlated multi-streaming", 21st International Workshop on Network and operating systems support for digital audio and video (NOSSDAV'11), Vancouver, British Columbia, Canada, pp. 81-86, June 2011.

[Aky96] I.F. Akyildiz, W. Yen, "Multimedia Group Synchronization Protocols for Integrated Services Networks", IEEE Journal On Selected Areas In Communications, Vol. 14, No. 1, January 1996.

[Alm99] G. Almes, S. Kalidindi, M. Zekauskas, "A One-way Delay Metric for IPPM", IETF Internet Standard, RFC 2679, September 1999.

[Asa14] H. Asaeda, R. Huang, Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Blocks for Synchronization Delay and Offset Metrics Reporting", IETF Internet Standard, RFC 7244, May 2014.

[Ban09] Y. Bang, J. Han, K. Lee, J. Yoon, J. Joung, S. Yang, and J-K. Kevin Rhee, "Wireless Network Synchronization for Multichannel Multimedia Services", Feb. 15-18, 2009 IEEE ICACT 2009.

[Bar07] I. Bartoli, G. Iacovoni, F. Ubaldi, "A Synchronization Control Scheme for Videoconferencing Services", Journal of Multimedia, Vol 2, No 4, pp. 1-9, August 2007.

[Beg09] Begen A. C., Glazebrook N., Ver Steeg W., "Reducing Channel-Change Times with the Real-Time Transport Protocol", IEEE Internet Computing, 13(3), pp. 40-47, June 2009.

[Beg10] A. Begen, C. Perkins, and J. Ött, "On the use of RTP for Monitoring and Fault Isolation in IPTV", IEEE Network, Volume 24 Issue 2, March/April 2010.

[Beg11a] A. Begen, T. Akgul, M. Baugher, "Watching Video over the Web: Part 1: Streaming Protocols", IEEE Internet Computing, vol.15, no.2, pp. 54-63, April 2011.

[Beg11b] A. Begen, T. Akgul, M. Baugher, "Watching Video over the Web: Part 2: Applications, Standardization, and Open Issues", IEEE Internet Computing, vol.15, no.3, pp. 59-63, June 2011.

[Beg12] A. Begen, "Considerations for Deploying the Rapid Acquisition of Multicast RTP Sessions (RAMS) Method", RFC 6659, July 2012.

[Bgc10] Begic Z., Kos, M., "Rapid synchronization of RTP multicast sessions", International Journal of Computer Science and Network Security, 10(9), pp. 42-47, September 2010.

[Bel10] L. Beloqui, H. Melvin, "Enhanced IPTV services through time synchronisation", 14th IEEE International Symposium on Consumer Electronics (ISCE) 2010, pp. 1-6, Braunschweig (Germany), June 2010.

[Bel12a] L. Beloqui, S. Al-Majeed, H. Melvin, M. Fleury, "Effective synchronisation of Hybrid Broadcast and Broadband TV", IEEE International Conference on Consumer Electronics (ICCE), 2012, pp. 160-161, Las Vegas (USA), January 2012.

[Bel12b] L. Beloqui, H. Melvin, "Client-side Multisource Media Streams Multiplexing for HbbTV", IEEE 2nd International Conference on Consumer Electronics (ICCE), 2012, pp. 1-5, Berlin (Germany), September 2012.

[Bel12c] L. Beloqui, P. O. Flaithearta, H. Melvin, "Interactive Multi-source Media Synchronisation for HbbTV", MediaSync Workshop 2012, Berlin (Germany), October 2012.

[Bie99] E. Biersack, W. Geyer, "Synchronized delivery and playout of distributed stored multimedia streams", ACM/Springer Multimedia Systems, Vol. 7 (1), pp. 70-90, January 1999.

[Bla96] G. Blakowski, R. Steinmetz, "A media synchronization survey: reference model, specification, and case studies", IEEE Journal on Selected Areas in Communications, 14(1), pp. 5-35, January 1996.

[Bou08] C. Bouras, A. Gkamas, and G. Kioumoutzis, "Extending the Funtionality of RTP/RTCP Implementation in Network Simulator (NS-2) to support TCP friendly congestion control", SIMUTools 2008, Marseille (France), March 2008.

[Bor08] F. Boronat, J. C. Guerri, and J. Lloret, "An RTP/RTCP based approach for multimedia group and inter-stream synchronization", Multimedia Tools and Applications Journal, Vol. 40 (2), pp. 285-319, June 2008.

[Bor09a] F. Boronat, J. Lloret, and M. García, "Multimedia group and inter-stream synchronization techniques: A comparative study", Information Systems, 34(1), pp. 108-131, March 2009.

[Bor09c] F. Boronat, M. **Montagud**, and J. C. Guerri, "Multimedia group synchronization approach for one-way cluster-to-cluster applications", IEEE 34th Conference on Local Computer Networks (IEEE LCN 2009), pp. 177-184, Zürich (Switzerland), October 2009.

[Bou09] C. Bouras, A. Gkamas, G. Kioumourtzis, "Evaluation of Single Rate Multicast Congestion Control Schemes for MPEG-4 Video Transmission", 5th Euro-NGI conference on Next Generation Internet networks (NGI'09), Aveiro (Portugal), July 2009.

[Bra13] R. van Brandenburg, A. Veenhuizen, "Immersive second-screen experiences using hybrid media synchronization", Media Sync Workshop 2013, Nantes (France), October 2013.

[Cal03] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[Car03] M. Carson, D. Santay, "NIST Net: a Linux-based network emulation tool. SIGCOMM Comput. Commun. Rev". 33, 3, 111-126, July 2003.

[Car09] V. Carrascal, RTP/RTCP code for NS-2, Grupo de Servicios Telemáticos, Universidad Politécnica de Cataluña, España. http://gridnet.upc.es/~maguilar/ns2_code_victor/ns2codevictor.

[Cas03] S. Casner, "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, July 2003.

[Ces09a] P. Cesar, D.C.A. Bulterman, J. Jansen, "Leveraging the User Impact: An Architecture for Secondary Screens Usage in an Interactive Television Environment", ACM/Springer Multimedia Systems, Vol. 15 (3), pp.127-142, June 2009.

[Ces09b] P. Cesar, D.C.A. Bulterman, J. Jansen, D. Geerts, H. Knoche, W. Seager, "Fragment, tag, enrich, and send: Enhancing social sharing of video", ACM Transactions on Multimedia Computing Communications and Applications (TOMCCAP). 5, 3, Article 19, 27 pages, August 2009.

[Ces14] P. Cesar, R. Kaiser, M. Ursu, "Toward Connected Shared Experiences", Computer, vol.47, no.7, pp.86-89, July 2014.

[Cha07] J. Cha, Y. Seo, Y. Kim; J. Ryu, "An Authoring/Editing Framework for Haptic Broadcasting: Passive Haptic Interactions using MPEG-4 BIFS", Second Joint Euro Haptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, pp. 274-279, Tsukuba (Japan), March 2007.

[Che03] M. Chen, "A low-latency lip-synchronized videoconferencing system", SIGCHI Conference on Human Factors in Computing Systems, ACM CHI'03, 464-471, New York, 2003.

[Chu07] H. Chuang, C. Huang, and T. Chiang, "Content-Aware Adaptive Media Playout Controls for Wireless Video Streaming", IEEE Transactions on Multimedia, Vol.9, No. 6, pp. 1273-1283, October 2007.

[Cla13a] A. Clark, K. Gross, Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay Metric Reporting", IETF Internet Standard, RFC 6843, January 2013.

[Cla13b] A. Clark, V. Singh, Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for De-Jitter Buffer Metric Reporting", IETF Internet Standard, RFC 7005, September 2013.

[Con12] C. Concolato, S. Thomas, R. Bouqueau, J. Le Feuvre, "Synchronized Delivery of Multimedia Content over Uncoordinated Broadcast Broadband Networks", ACM Multimedia Systems Conference, MMSys 2012, Chapel Hill, North Carolina (USA), February 2012.

[Con13] C. Concolato, J. Le Feuvre, "Live HTTP Streaming of Video and Subtitles within a Browser", ACM Multimedia Systems Conference, MMSys 2013, Oslo (Norway), February 2013.

[Cro04] E. Cronin, B. Filstrup, S. Jamin, A.R. Kurc, An efficient synchronization mechanism for mirrored game architectures, Multimedia Tools and Applications Journal, 23(l), pp. 7–30, 2004.

[Dev08] M. O. Van Deventer, H. Stokking, O. A. Niamut, F. A. Walraven, V. B. Klos, "Advanced Interactive Television Service Require Synchronization", 15th International Conference on Systems, Signals and Image Processing, IWSSIP 2008, Bratislava (Slovak Republic), June 2008.

[Dio99] C. Diot and L. Gautier, "A Distributed Architecture for Multiplayer Interactive Applications on the Internet", IEEE Network, vol. 13, nº 4, pp. 6-15, July/August 1999.

[Dom04] H.P. Dommel, S.K. Verma, "Multipoint synchronization protocol", IEEE International Conference on Systems, Man and Cybernetics, pp. 4631-4635, The Hague (Netherlands), October 2004.

[Ehl94] L. Ehley, B. Furht, M. Ilyas, "Evaluation of multimedia synchronization techniques", International Conference on Multimedia Computing and Systems, pp. 514-519, May 1994.

[ETSI TS 181 061]

ETSI TS 181 016 V3.3.1, "Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Service Layer Requirements to integrate NGN Services and IPTV", July 2009.

[ETSI TS 182 027]

ETSI TS 182 027 V3.5.1 (2011-03), "Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); IPTV Architecture; IPTV functions supported by the IMS subsystem", March 2011.

[ETSI TS 183 063]

ETSI TS 183 063 V3.5.2, "Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); IMS-based IPTV stage 3 specification", March 2011.

[Fer10] F. Ferrari, A. Meier, and L. Thiele, "Accurate Clock Models for Simulating Wireless Sensor Networks", SIMUTools 2010, Torremolinos (Spain), March 2010.

[Fle08] R.D.S. Fletcher.; "Consistency Maintenance for Multiplayer Video Games", MsC, Queens University (Canada), January 2008.

[Fle10] C. Fleury, T. Duval, V. Gouranton, B. Arnaldi, "Architectures and Mechanisms to Efficiently Maintain Consistency in Collaborative Virtual Environments", SEARIS 2010, Massachussets (USA), March 2010.

[Fri03] Friedman T., Caceres R., Clark A., "RTP Control Protocol Extended Report (RTCP XR)", IETF Internet Standard, RFC 3611, November 2003.

[Fuc08] H. Fuchs, N. Farber, "Optimizing channel change time in IPTV applications", IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (IEEE BMSB 2008), Las Vegas (USA), April 2008.

[Gee08] D. Geerts, P. Cesar, D. Bulterman, "The implications of program genres for the design of social television systems", 1st international conference on Designing interactive user experiences for TV and video (UXTV'08), Silicon Valley, California (USA), October 2008.

[Gee09] D. Geerts, D. De Grooff, "Supporting the social uses of television: sociability heuristics for social TV", ACM CHI Conference on Human Factors in Computing Systems, CHI 2009, pp. 595-604, Boston (USA), April 2009.

[Gee11] D. Geerts, I.Vaishnavi, R. Mekuria, O. van Deventer, and P. Cesar, "Are we in sync?: synchronization requirements for watching online video together", ACM CHI Conference on Human Factors in Computing Systems, CHI 2011, Vancouver (Canada), May 2011.

[Gue01] J. C. Guerri, M. Esteve, C. E. Palau, and V. Casares, "Feedback Flow Control with Hysteresial Techniques for Multimedia Retrievals", Multimedia Tools and Applications, 13, 3, pp. 307-332, March 2001.

[Gro14] K. Gross, R. van Brandenburg, "RTP and Leap Seconds", IETF Internet Standard, RFC 7164, March 2014.

[Gro03] C. Groves, M. Pantaleo, T. Anderson, and T. Taylor, "Gateway Control Protocol Version 1", RFC 3525, June 2003.

[Han06] M. Handley, V. Jacobson, C. Perkins, "SDP: Session Description Protocol", IETF Internet Standard, RFC 4566, July 2006.

[Haas Effect]

Haas Effect, Wikipedia, http://en.wikipedia.org/wiki/Precedence_effect, Last Accessed in November 2014.

[Has06] T. Hashimoto, Y. Ishibashi, "Group Synchronization Control over Haptic Media in a Networked Real-Time Game with Collaborative Work", Netgames'06, Singapore, October 2006.

[Hes10] C. Hesselman, D. Abbadessa, W. Van Der Beek, et alter, "Sharing enriched multimedia experiences across heterogeneous network infrastructures", IEEE Communications Magazine, Vol.48, no.6, pp.54-65, June 2010.

[Hol14] C. Holmberg, S. Hakansson, G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", draft-ietf-rtcweb-use-cases-and-requirements-14, IETF RTCWEB Working Group, Internet Draft, February 2014.

[Hos09] Hosoya K., Ishibashi. Y, Sugawara S., Psannis K.E., "Group synchronization control considering difference of conversation roles," IEEE 13th International Symposium on Consumer Electronics (ISCE '09). pp. 948-952, Kyoto (Japan), May 2009.

[Hss11] T. Hossfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, R. Schatz, "Quantification of YouTube QoE via Crowdsourcing", IEEE International Symposium on Multimedia (ISM) 2011, December 2011.

[How11] C. Howson, E. Gautier, "Second screen TV synchronization", International Conference on Consumer Electronics, pp. 361–365, Berlin (Germany), September 2011.

[Hua09] E. M. Huang, G. Harboe, J. Tullio, A. Novak, N. Massey, C.J. Metcalf, G. Romano, "Of social television comes home: a field study of communication choices and practices in tv-based text and voice chat", ACM CHI Conference on Human Factors in Computing Systems, CHI 2009, pp. 585-594, Boston (USA), April 2009.

[Hua11] Z. Huang, W. Wu, K. Nahrstedt, R. Rivas, A. Arefin, "SyncCast: synchronized dissemination in multi-site interactive 3D tele-immersion", 2nd annual ACM conference on Multimedia systems (MMSys 2011), San Jose, California (USA), February 2011.

[Hua12] Z. Huang, "Synchronized Distribution Framework for High-Quality Multi-modal Interactive Teleimmersion", PhD Thesis, University of Illinois at Urbana-Champaign, Supervisor: K. Nahrstedt, 2012.

[Hua13] Z. Huang, K. Nahrstedt, R. Steinmetz, "Evolution of temporal multimedia synchronization principles: A historical viewpoint", ACM Transactions on Multimedia Computing Communications and Applications (TOMCCAP). 9, 1s, Article 34, 23 pages, October 2013.

[Hua14] P. Huang and Y. Ishibashi, "QoE assessment of will transmission using vision and haptics in networked virtual environment", International Journal of Communications, Network and System Sciences (IJCNS), vol. 7, no. 8, pp. 265-278, August 2014.

[IEEE 1588]

IEEE 1588 Precision Time Protocol (PTP) Version 2 Specification, IEEE Standard, March 2008.

[Ish97a] Y. Ishibashi, A. Tsuji, and S. Tasaka, "A Group Synchronization Mechanism for Stored Media in Multicast Communications", Proc. of the INFOCOM '97, Washington April 1997.

[Ish97b] Y. Ishibashi, and S. Tasaka, "A group synchronization mechanism for live media in multicast communications", IEEE GLOBECOM'97, pp. 746–752, November 1997.

[Ish99] Y. Ishibashi, and S. Tasaka, "A distributed control scheme for group synchronization in multicast communications", Proc. of International Symposium Communications, Kaohsiung (Taiwan), pp. 317-323, November 1999.

[Ish00] Y. Ishibashi, S. Tasaka, "A comparative survey of synchronization algorithms for continuous media in network environments", 25th Annual IEEE Conference on Local Computer Networks, LCN 2000, pp.337-348, Florida (USA), November 2000.

[Ish02] Y. Ishibashi, and S. Tasaka, "A distributed control scheme for causality and media synchronization in networked multimedia games", Proc. 11th International Conference on Computer Communications and Networks, Miami (USA), pp. 144-149, October 2002.

[Ish03a] Y. Ishibashi, K. Tomaru, S. Tasaka, and K. Inazumi, "Group synchronization in networked virtual environments", Proc. Of the 38th IEEE International Conference on Communications, Alaska (USA), pp. 885–890, May 2003.

[Ish03b] Y. Ishibashi, S. Tasaka, H. Ogawa, "Media Synchronization Quality of Reactive Control Schemes", IEICE Transactions on Communications, Vol.E86-B, No.10, pp. 3103-3113, October 2003.

[Ish04] Y. Ishibashi, T. Hasegawa, and S. Tasaka, "Group synchronization control for haptic media in networked virtual environments", Proc. 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Chicago (USA), pp. 106-113, March 2004.

[ITU-T G.114]

ITU-T Rec. G. 114: "One-way transmission time", 2003.

[ITU-T G.1050]

ITU-T Rec. G.1050: "Network model for evaluating multimedia transmission performance over Internet Protocol", 2007.

[ITU-T G.1010]

ITU-T Rec. G.1010: "End-user multimedia QoS categories", 2001.

[ITU-T P.862]

ITU-T Rec. P.862: "P.862: Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs", 2001.

[Jan11] J. Jansen, P. Cesar, D.C.A. Bulterman, T. Stevens, I. Kegel, J. Issing, "Enabling Composition-Based Video-Conferencing for the Home", IEEE Transactions on Multimedia (TMM), 13(5), pp. 869-881, October 2011.

[Jan13a] J. Jansen, D.C.A. Bulterman, "User-centric video delay measurements", 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '13), pp. 37-42, Oslo (Norway), February 2013.

[Jan13b] J. Jansen, P. Cesar, D.C.A. Bulterman, "Multimedia Document Synchronization in a Distributed Social Context". ACM Symposium on Document Engineering (DocEng 2013), Florence (Italy), pp. 273–276, September 2013.

[Jen13] C. Jennings, T. Hardie, M. Westerlund, "Real-time communications for the web", IEEE Communications Magazine, vol.51, no.4, pp. 20-26, April 2013.

[Joh09] I. Johansson, M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", IETF Internet Standard, RFC 5506, April 2009.

[Kal04] M. Kalman, E. Steinbach, B. Girod, "Adaptive media playout for low-delay video streaming over error-prone channels," IEEE Trans. Circuits Syst. Video Technol., 14(6), pp. 841–851, June 2004.

[Kao06] K. Kao, C. Ke, C. Shieh, "An advanced simulation tool-set for video transmission performance evaluation", The 2006 Workshop on Ns-2: the IP Network Simulator (WNS2'06), Pisa (Italy), October 2006.

[Ke08] C.H. Ke, C.K. Shieh, W.S. Hwang, A. Ziviani, "An Evaluation Framework for More Realistic Simulations of MPEG Video Transmission", Journal of Information Science and Engineering, 24(2), pp. 425-440, 2008.

[Kla03] J. Klaue, B. Rathke, A. Wolisz, "EvalVid – A Framework for video transmission and quality evaluation", 13th International Conference on Modelling, Techniques and Tools for Computer Performance Evaluation, Urbana, Illinois (USA), September 2003.

[Kle09] A. Klein, J. Klaue, "Performance Evaluation Framework for Video Applications in Mobile Networks", IEEE MESH 2009, Athens (Greece), June 2009.

[Ker10] R. Kernchen, S. Meissner, K. Moessner, P. Cesar, I. Vaishnavi, M. Boussard, C. Hesselman, "Intelligent Multimedia Presentation in Ubiquitous Multidevice Scenarios", IEEE Multimedia, vol.17, no.2, pp.52-63, June 2010.

[Kim05] S.J. Kim, F. Kuester, K. Kim, "A global timestamp-based approach for enhanced data consistency and fairness in collaborative virtual

environments", ACM/Springer Multimedia Systems Journal, Vol. 10 (3), pp. 220-229, February 2005.

[Koo14] W. Kooij, "Playout Delay of TV Broadcasting", Master Thesis, University of Twente & TNO, 2014.

[Köh94] D. Köhler, H. Müller, "Multimedia playout synchronization using buffer level control, Second International Workshop on Advanced Teleservices and High-Speed Communication Architectures, pp.167–180, Heidelberg (Germany), September 1994.

[Kry13] A. Kryczka, A. Arefin, K. Nahrstedt, "AvCloak: A Tool for Black Box Latency Measurements in Video Conferencing Applications", (ISM), IEEE International Symposium on Multimedia (IEEE ISM 2013), pp. 271-278, Anaheim, California (USA) December 2013.

[Kro04] Kropfberger, M., Hellwagner, H., Evaluation of RTP immediate feedback and retransmission extensions, ICME 2004, 1751-1754, Taipei (Taiwan), June 2004.

[Kuo01] C.C. Kuo, M.S. Chen, J.C. Chen, "An Adaptive transmission scheme for audio and video synchronization based on real-time transport protocol", IEEE International Conference on Multimedia and Expo, ICME 2001, pp. 403-406, Tokyo (Japan), August 2001.

[Kur07] Y. Kurokawa, Y. Ishibashi, and T. Asano, "Group synchronization control in a remote haptic drawing system", Proc. of IEEE International Conference on Multimedia and Expo, Beijing (China), pp. 572-575, July 2007.

[Lao02] N. Laoutaris, and I. Stavrakakis, "Intrastream synchronization for continuous media streams: a survey of playout schedulers", IEEE Network Magazine, 16 (3), 30-40, 2002.

[Len09] J. Lennox, J. Ott, T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", IETF Internet Standard, RFC 5576, June 2009.

[Len14] J. Lennox, M. Westerlund, Q. Wu, C. Perkins, "Sending Multiple Media Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-06, IETF AVTCORE Group, Internet Draft, October 2014.

[Ler07] P. Leroux, V. Verstraete, F. De Turck, P. Demeester, "Efficient management of synchronised interactive services through the design of MCDP middleware", Australasian Telecommunication Networks and Applications Conference (ATNAC 2007), pp. 215-220, Christchurch (New Zealand), December 2007.

[Li08] Y. Li, A. Markopoulou, J. Apostolopoulos, N. Bambos, "Content-Aware Playout and Packet Scheduling for Video Streaming Over Wireless Links", IEEE Trans. on Multimedia, 10(5), pp. 885–895, August 2008.

[Li12] M. Li, T. Lin, S. Cheng, "Arrival process-controlled adaptive media playout with multiple thresholds for video streaming", Multimedia Systems, 18(5), pp. 391-407, October 2012.

[Li13] B. Li, Z. Wang, J. Liu, W. and Zhu, "Two decades of Internet video streaming: A retrospective view", ACM Transactions on Multimedia Computing Communications and Applications (TOMCCAP). 9, 1s, Article 33, 20 pages, October 2013.

[Lie08] A. Lie, J. Klaue, "Evalvid-RA: Trace Driven Simulation of Rate Adaptive MPEG-4 VBR Video", Multimedia Systems, 14(1), pp. 33-50, June 2008.

[Lit91] T. D. C. Little and A. Ghafoor, "Spatio-temporal composition of distributed multimedia objects for value-added networks", IEEE Computer, vol. 24, no. 10, pp. 42–50, October 1991.

[Liu09] C. Liu, C., et al. C. Liu, Y-K. Wang, M.M. Hannuksela, C. Ying, M. Sujeet, M. Gabbouj, "RTP/AVPF compliant feedback for error resilient video coding in conversational applications", 9th International Symposium on Communications and Information Technology, ISCIT 2009, pp. 218-223, Incheon (Korea), September 2009.

[Lom97] M. Lombard, T. Ditton, "At the Heart of It All: The Concept of Presence, Journal of Computer-Mediated Communication", September 1997.

[Lor12] S. Loreto, S.P. Romano, "Real-Time Communications in the Web: Issues, Achievements, and Ongoing Standardization Efforts", IEEE Internet Computing, vol.16, no.5, 68-73, October 2012.

[Lu09a] Z. Lu, C. Jin-Zhong; Z. Ji-Xian, "A Method for Storage and Transport of Embedded Rich Media Application", International Symposium on

Computer Network and Multimedia Technology (CNMT 2009), pp. 1-4, Wuhan (China), January 2009.

[Luk03] S. Lukosch, "Transparent Latecomer Support for Synchronous Groupware", 9th International Workshop on Groupware (CRIWG 2003), pp. 26-41, Grenoble (France), September 2003.

[Mai09] J. Maisonneuve, M. Deschanel, J. Heiles, L. Wei, L. Hong, R. Sharpe, W. Yiyan, "An Overview of IPTV Standards Development", IEEE Transactions on Broadcasting, 55(2), pp. 315-328, June 2009.

[Mag09] F.P. Margalef, A. Lopez, G. Fernandez, "Frame-accuracy synchronisation for mobile TV interactivity", IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, (BMSB'09), Bilbao (Spain), pp. 1-5, May 2009.

[Man06] S.S. Manvi, P. Venkataram, "Agent-based subsystem for multimedia communications", IEE Proceedings - Software, vol.153, no.1, pp. 38-48, February 2006.

[Man13] D.A.G. Manzato, N.L.S. da Fonseca, "A survey of channel switching schemes for IPTV", IEEE Communications Magazine, 51(8), pp. 120-127, August 2013.

[Mar09] M. C. Marques Neto, C. A. S. Santos, "An approach based on events for treating the late tuning problem in interactive live TV shows" 1st ACM international workshop on Events in multimedia (EiMM'09), co-located with ACM Multimedia Conference, Beijing (China), October 2009.

[Mat00] S. Matsumoto, I. Fukuda, H. Morino, K. Hikichi, K. Sezaki, and Y. Yasuda, "The influences of network issues on haptic collaboration in shared virtual environments," 5th Phantom Users' Group Workshop (PUG'00), Aspen, Colo, USA, October 2000.

[Mau04] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg, "Local-Lag and Timewarp: Providing Consistency for Replicated Continuous Applications", IEEE Transactions on Multimedia, Vol.6, No.1, February 2004.

[Mek11] R. Mekuria, "Inter-destination media synchronization for TV broadcasts", Master Thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, Dept. of Network architecture and Services, Delft University of Technology, April 2011.

[Mek12] R. Mekuria, P. Cesar, D. Bulterman, "Digital TV: the effect of delay when watching football", 10th European conference on Interactive TV and video (EuroiTV '12), Berlin (Germany), July 2012.

[Men14] R. Mendes Costa Segundo, C. A. Saibel Santos, "Systematic Review of Multiple Contents Synchronization in Interactive Television Scenario", ISRN Communications and Networking, Volume 2014 (2014), Article ID 127142, pp. 1-17, February 2014.

[Mey93] T. Meyer, W. Effelsberg, R. Steinmetz, "A taxonomy on multimedia synchronization", Fourth Workshop on Future Trends of Distributed Computing Systems, pp. 97-103, September 1993.

[Mil90] D. Mills, "On the accuracy and stablility of clocks synchronized by the network time protocol in the internet system", SIGCOMM Comput. Commun. Rev., 20(1), pp.65–75, 1990.

[Mil10] D. Mills, U. Delaware, J. Martin, J. Burbank, W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", IETF Internet Standard, RFC 5905, June 2010.

[Min99] N. Minar, "A survey of the NTP network", http://www.media.mit.edu/wnelson/research/ntp-survey99/, December 1999.

[Miy11] Y. Miyashita, Y. Ishibashi, N. Fukushima, S. Sugawara, K. E. Psannis, "QoE assessment of group synchronization in networked chorus with voice and video", IEEE Region 10 Conference (IEEE TENCON'11), pp. 393-397, Bali (Indonesia), November 2011.

[Mtp11] M. Montpetit, N. Klym, T. Mirlacher, "The future of IPTV - Connected, mobile, personal and social", Multimedia Tools and Applications Journal, Vol. 53 (3), pp. 519-532, July 2011.

[MPEG] Moving Picture Experts Group (MPEG), http://mpeg.chiariglione.org/.

[Mur13] N. Murray, Y. Qiao, B. Lee, A.K. Karunakar, G-M. Muntean, "Subjective evaluation of olfactory and visual media synchronization", ACM Multimedia Systems Conference (MMSys 2013), 162-171, Oslo (Norway), February 2013.

[Mur14] N. Murray, Y. Qiao, B. Lee, G.-M. Muntean, "User-profile-based perceived olfactory and visual media synchronization", ACM Trans.

Multimedia Comput. Commun. Appl. (ACM TOMCCAP Journal), 10, 1s, Article 11, 24 pages, January 2014.

[Mrt06] C. Murta, P. Torres Jr, and P. Mohapatra, "Characterizing quality of time and topology in a time synchronization network". 49th IEEE Global Telecommunications Conference, IEEE GLOBECOM 2006, San Francisco, CA, USA, November 2006.

[NCL] NCL (Nested Context Language), http://www.ncl.org.br/en/inicio.

[Nie93] J. Nielsen, Response Times: The Three Important Limits, January 1993, Available: http://www.useit.com/papers/responsetime.html

[Num05] T. Nunome, and S. Tasaka, "Inter-destination synchronization quality in a multicast mobile ad hoc network", Proc. of IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, Berlin (Germany), pp. 1366-1370, September 2005.

[Ott06] J. Ott, S. Wenger, N. Sato, C. Burmeister, J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", IETF Internet Standard, RFC 4585, July 2006.

[Ott07] D.E. Ott, K. Mayer-Patel, "An open architecture for transport-level protocol coordination in distributed multimedia applications", ACM Transactions on Multimedia Computing Communications and Applications (TOMCCAP). 3, 3, Article 17, August 2007.

[Ott10a] J. Ott, J. Chesterfield, E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", IETF Internet Standard, RFC 5760, February 2010.

[Ott10b] J. Ott, and C. Perkins, "Guidelines on Extending the RTP Control Protocol (RTCP)", IETF Internet Standard, RFC 5968, September 2010.

[Pal04] C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, "On maintaining interactivity in event delivery synchronization for mirrored game architectures", IEEE Global Telecommunications Conference Workshops (GlobeCom Workshops 2004), pp. 157-165, December 2004.

[Par08] S. Park, J. Kim, "An adaptive media playout for intra-media synchronization of networked-video applications", Journal of Visual Communications & Image Representation, Vol. 19, 106-120, 2008.

[Per96] M.J. Perez-Luque, T.D.C. Little, "A temporal reference framework for multimedia synchronization", IEEE Journal on Selected Areas in Communications, 14(1), pp. 36-51, January 1996.

[Per10a] C. Perkins, M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", IETF Internet Standard, RFC 5761, April 2010.

[Per10b] C. Perkins, T. Schierl, "Rapid Synchronization of RTP Flows", IETF Internet Standard, RFC 6051, November 2010.

[Per14] C. Perkins, M. Westerlund, J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-16, IETF RTCWEB Working Group, Internet Draft, July 2014.

[Pre10] W. Premchaiswadi, A. Tungkasthan, N. Jongsawat, "Enhancing learning systems by using virtual interactive classrooms and web-based collaborative work", IEEE Education Engineering Conference (EDUCON 2010), pp 1531-1537, Madrid (Spain), April 2010.

[Pit10] Pitt, Ian, CS2511 Usability Engineering Lecture Notes, Localisation of Sound Sources, http://web.archive.org/web/20100410235208/http:/www.cs.ucc.ie/~ianp/CS2511/HAP.html, Last Access in October 2014.

[Ram11] F. Ramos, J. Crowcrofta, R.J. Gibbensa, P. Rodriguez, I.H. White, "Reducing channel change delay in IPTV by predictive pre-joining of TV channels", Signal Processing: Image Communication, 26(7), pp.400-412, August 2011.

[Ram13] F. Ramos, "Mitigating IPTV zapping delay", IEEE Communications Magazine, 51(8), pp. 128-133, August 2013.

[Ran95] P. V. Rangan, S. Ramanathan, and S. Sampathkumar, "Feedback techniques for continuity and synchronization in multimedia information retrieval", ACM Trans. Inf. Syst, 13, 2, 145-176, April 1995.

[Rai04] B. Rainer, C. Timmerer, "A subjective evaluation using crowdsourcing of Adaptive Media Playout utilizing audio-visual content features", IEEE NOMS, pp. 1-7, May 2014.

[Rid10] J. Ridoux, and R.Veitch, "Principles of Robust Timing over the Internet", Com. of the ACM, Vol. 53 (No. 5), May 2010.

[Riv13] R. Rivas, "StreamOS: Distributed Operating System for Correlated Multi-modal Streaming Applications", PhD Thesis, University of Illinois at Urbana-Champaign, Supervisor: K. Nahrstedt, 2013.

[Roc08] M. Roccetti; S. Ferretti; C. Palazzi; "The Brave New World of Multiplayer Online Games: Synchronization Issues with Smart Solution", 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC), pp. 587-592, Orlando, Florida (USA), May 2008.

[Rod12] A. Rodriguez-Alsina, G. Talavera, P. Orero and J. Carrabina, "Subtitle Synchronization across Multiple Screens and Devices", Sensors 2012, 12(7), pp. 8710-8731, June 2012.

[Rom14a] A. Romanov, S. Botzko, M. Duckworth, R. Even, "Use Cases for Telepresence Multistreams", IETF Internet Standard, RFC 7205, April 2014.

[Rom14b] A. Romanov, S. Botzko, M. Barnes, "Requirements for Telepresence Multi-Streams", IETF Internet Standard, RFC 7262, June 2014.

[Ros02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M Handley, E. Schooler, "SIP: Session Initiation Protocol", IETF Internet Standard, RFC 3261, June 2002.

[Sch13] M. Schmitt, S. Gunkel, D.C.A. Bulterman, P. Cesar, "A Quality of Experience Testbed for video-mediated group communication", IEEE International Symposium on Multimedia (ISM 2013), Anaheim, California (USA), pp. 514-515, December 2013.

[Sch93] E. Schooler, "Distributed Music: A Foray into Networked Performance", International Network Music Festival, Santa Monica (USA), September 1993.

[Sch98] H. Schulzrinne, A. Rao, R. Lanphier, "Real-Time Streaming Protocol (RTSP)", RFC 2326, IETF Internet Standard, April 1998.

[Sch03a] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", IETF Internet Standard, RFC 3550, July 2003.

[Sch03b] H. Schulzrinne, S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", IETF Internet Standard, RFC 3551, July 2003.

[Sha12a] U.D. Shahab, D. Bulterman, "Synchronization Techniques in Distributed Multimedia Presentation", The Fourth International Conferences on Advances in Multimedia (MMEDIA 2012), Chamonix / Mont Blanc (France), April 2012.

[Sha12b] U.D. Shahab, "An Architecture and Implementation for Evaluating Synchronization Support for Shared User Experiences", MsC Thesis, Vrije Universiteit (VU) Amsterdam (Netherlands), CWI, Supervisors: D. Bulterman and J. Jacksen, October 2012.

[Sha08] Shamma, D.A., Bastea-Forte, M., Joubert, N., Liu, Y.: "Enhancing online personal connections through synchronized sharing of online video", The 26th Annual CHI Conference on Human Factors in Computing Systems, ACM CHI'08 Extended Abstracts, Florence (Italy), April 2008.

[SMIL] Synchronized Multimedia Integration Language (SMIL) http://www.w3.org/AudioVideo/

[Sri97] M. A. Srinivasan, C. Basdogan, "Haptics in virtual environments: Taxonomy, research status, and challenges", Computers and Graphics, vol. 21, no. 4, pp. 393–404, April 1997.

[Ste11] B. Ver Steeg, A. Begen, T. Van Caenegem, Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC6285, June 2011.

[Ste96] R. Steinmetz, "Human Perception of Jitter and Media Synchronization", IEEE Journal On Selected Areas In Communications, Vol.14, No.1, 61-72, January 1996.

[Sto10] H. Stokking, O. van Deventer, O. Niamut, F. Walraven, and R. Mekuria, "IPTV inter-destination synchronization: A network-based approach", ICIN'2010, Berlin, October 2010.

[Stoc11] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles", Second annual ACM conference on Multimedia systems, MMSys 2011, San Jose – California (USA), pp. 133-144, February 2011.

[Su09] Y. Su, Y. Yang, M. Lu, H. Chen, "Smooth Control of Adaptive Media Playout for Video Streaming", IEEE Transactions on Multimedia, Vol.1, No. 7, 1331-1339, November 2009.

[Swa13] V. Swaminathan, "Are we in the middle of a video streaming revolution?", ACM Transactions on Multimedia Computing Communications and Applications (TOMCCAP). 9, 1s, Article 40, 6 pages, October 2013.

[Tas98] S. Tasaka, Y. Ishibashi, "A Performance Comparison of Single-Stream and Multi-Stream Approaches to Live Media Synchronization", IEICE Trans. Commun., vol.E81-B, no.11, pp. 1988-1997, November 1998.

[Tas00] S. Tasaka, T. Nunome, Y. Ishibashi, "Live media synchronization quality of a retransmission-based error recovery scheme", IEEE International Conference on Communications, New Orleans, LA, USA, pp. 1535–1541, June 2000.

[Tas02] S. Tasaka, Y. Ishibashi, M. Hayashi, "Inter–destination synchronization quality in an integrated wired and wireless network with handover", IEEE GLOBECOM 2002, pp. 1560- 1565 vol.2, Nov. 2002.

[Tim14] C. Timmerer, B. Rainer, "The Social Multimedia Experience", Computer, vol.47, no. 3, pp. 67-69, March 2014.

[Vai11a] I. Vaishnavi, "Coherence in Synchronous Shared Experiences", PhD Thesis, Vrije Universiteit (VU) Amsterdam (Netherlands), CWI, Supervisors: D. Bulterman and P. Cesar, June 2011.

[Vai11b] I. Vaishnavi, P. Cesar, D. Bulterman, O. Friedrich, S. Gunkel, D. Geerts, "From IPTV to synchronous shared experiences challenges in design: Distributed media synchronization", Signal Processing: Image Communication, Vol. 26, Issue 7, pp. 370-377, August 2011.

[Vee12] A. Veenhuizen, R. van Brandenburg, "Frame accurate media synchronization of heterogeneous media sources in an HBB context", Media Sync Workshop 2012, Berlin (Germany), October 2012.

[Xie99] Y. Xie, C. Liu, M.J. Lee, T.N. Saadawi, "Adaptive multimedia synchronization in a teleconference system", ACM/Springer Multimedia Systems, Vol. 7 (4), pp. 326-337, July 1999.

[Wij12a] M. Wijnants, J. Dierckx, P. Quax, W. Lamotte, "synchronous MediaSharing: social and communal media consumption for geographically dispersed users", 3rd Multimedia Systems Conference (MMSys 2012), pp. 107-112, Chapel Hill, North Carolina (USA), February 2012.

[Wij12b] M. Wijnants, W. Lamotte, J. De Meulenaere, and W. Van den Broeck. "Qualitative assessment of contemporary media sharing practices and their relationship to the sMS platform", 1st International workshop on Socially-aware multimedia (SAM 2012), Nara (Japan), October 2012.

[Wil14] A. Williams, K. Gross, R. van Brandenburg, H. Stokking, "RTP Clock Source Signalling", RFC 7273, June 2014.

[Yas05] K. Yasumoto, K. Nahrstedt, "Ravitas: Realistic Voice Chat Framework for Cooperative Virtual Spaces", IEEE International Conference on Multimedia and Expo (ICME 2005), pp. 1046-1049, Amsterdam (The Netherlands), July 2005.

[Yu08] C-Y. Yu, C-H. Ke, R-S. Chen, C-K. Shieh, B. Munir, N. K. Chilamkurti, "MyEvalvid_RTP: A evaluation framework for more realistic simulations of multimedia transmissions", International Journal of Software Engineering and Its Applications, 2(2), April 2008.

[Yun10] J. Yun, J. Jonghyun, P. KwangRo, "Real-sense media synchronization technology based on the SMMD"," IEEE International Conference on Consumer Electronics (ICCE) 2010, pp. 71-72, Las Vegas (USA), January 2010.

[Zim08] R. Zimmermann, K. Liang, "Spatialized audio streaming for networked virtual environments", ACM International Conference on Multimedia (ACM MM 2008), pp. 299–308, Vancouver, British Columbia (Canada), October 2008

APPENDIX A PUBLICATIONS

Summary

This Appendix lists the publications derived from the research work in this PhD thesis. In numbers and categories, the publications are: 7 papers in International Journals (5 of them with J.C.R. Impact Factor), 10 papers in International Conferences (3 in Rank A and 2 in Rank A+ of the CORE Conference Ranking list), 2 papers in national conferences (JITEL), 3 papers in International Workshops, 2 posters, 1 Book Chapter, 1 RFC (IETF Standard), 2 IETF Internet Drafts and 1 demo in an international conference. In addition, our standardization works have been presented at several IETF meetings (concretely at 80th, 82th, 83th, 84th and 91th meetings), and will also be presented at the next one (92th) in March 2015.

From the 20 publications that required presentation, I was the speaker in 14 of them. Likewise, from the experience I have acquired attending conferences, and the researchers I have met there, I have also become Technical Program Committee (TPC) member of relevant international conferences (and close related to the topic of this PhD thesis), such as IEEE Local Computer Networks (LCN), ACM Multimedia and ACM TVX, as well as co-organizer of the international Media Synchronization (MediaSync) Workshop series.

The list of my publications and their citations can also be accessed via my Google Scholar profile.

List of Publications

[Bel15] J. Belda, **M. Montagud**, F. Boronat, M. Martínez, "Diseñando iWebSync: Una Plataforma Web 2.0 para la Sincronización Distribuida de Contenidos Multimedia e Interacción Social", Comunica2: Quinta Edición del Congreso Internacional Sobre Redes Sociales, Grau de Gandia (Spain), February 2015.

[Bor09b] F. Boronat, **M. Montagud**, "Ampliación de la funcionalidad de la implementación de RTP/RTCP en el simulador NS-2 para soportar la sincronización de grupo multimedia en aplicaciones Cluster-to-Cluster", VIII Jornadas de Ingeniería Telemática (JITEL 2009), Cartagena (Spain), September 2009.

[Bor10] F. Boronat, **M. Montagud**, V. Vidal, "Enhanced RTP-based tool-set for video streaming simulation using NS-2", 8th International Conference on Advances in Mobile Computing & Multimedia (MoMM 2010), pp. 382-386, Paris (France), November 2010.

[Bor11a] F. Boronat, **M. Montagud**, and V.Vidal, "A More Realistic RTP/RTCP-Based Simulation Platform for Video Streaming QoS Evaluation", J. Mobile Multimedia, 7 (1&2), pp. 66-88, April 2011.

[Bor11b] F. Boronat, **M. Montagud**, V. Vidal, "Master Selection Policies for Inter-Destination Multimedia Synchronization in Distributed Applications", IEEE 19th International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS 2011), Singapore, July 2011.

[Bor11c] F. Boronat, **M. Montagud**, "Propuesta de Sincronización Inter-Destinatario Adaptativa para Aplicaciones Multimedia Distribuidas", X Jornadas de Ingeniería Telemática (JITEL 2011), Santander (Spain), September 2011.

[Bor11d] F. Boronat, **M. Montagud**, and V.Vidal, "Smooth Control of Adaptive Media Playout to Acquire IDMS in Cluster-based Applications", IEEE 36th Conference on Local Computer Networks IEEE LCN 2011), pp. 617-625, Bonn (Germany), October 2011.

[Bor12a] F. Boronat, **M. Montagud**, H. Stokking, P. Cesar, R. Mekuria, "Future Issues and Challenges in Distributed Media Synchronization", MediaSync Workshop 2012, Berlin (Germany), October 2012.

[Bor12b] F. Boronat, **M. Montagud**, H. Stokking, O. Niamut, "The Need of Inter-Destination Synchronization for Emerging Social Interactive Multimedia Applications", IEEE Communications Magazine, **I.F. (2011): 3.785**, 50(11), pp. 150-158, November 2012.

[Bor13] F. Boronat, R. Mekuria, **M. Montagud**, P. Cesar, "Distributed Media Synchronization for Shared Video Watching: Issues, Challenges, and

Examples, Social Media Retrieval", Springer Computer Communications and Networks series, ISBN 978-1-4471-4554-7, 2013.

[Bra14] R. van Brandenburg, H. Stokking, O. Van Deventer, F. Boronat, **M. Montagud**, K. Gross, "Inter-destination Media Synchronization using the RTP Control Protocol (RTCP)", IETF Internet Standard, RFC 7272, June 2014.

[Mon10a] **M. Montagud**, F. Boronat, "A new network simulator 2 (NS-2) module based on RTP/RTCP protocols to achieve multimedia group synchronization", SIMUTOOLS 2010, Torremolinos, Spain, March 2010.

[Mon10b] **M. Montagud**, F. Boronat, V. Vidal, "Simulation platform for video streaming evaluation", 18th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2010), pp.397-401, September 2010.

[Mon11a] **M. Montagud**, F. Boronat, "On the Use of Adaptive Media Playout for Inter-Destination Synchronization", IEEE Communications Letters, **I.F. (2010): 1.060**, 15(8), pp. 863-865, August 2011.

[Mon11b] **M. Montagud**, F. Boronat, "Implementation and Evaluation of an M/S Scheme for Inter-Destination Multimedia Synchronization (IDMS)", Network Protocols and Algorithms, Vol 3 (3), pp. 80-98, December 2011.

[Mon12a] **M. Montagud**, F. Boronat, "Enhanced Adaptive RTCP-based Inter-Destination Multimedia Synchronization Approach for Distributed Applications", Computer Networks, **I.F. (2011): 1.2**, 56(12), pp. 2912-2933, August 2012.

[Mon12b] **M. Montagud**, F. Boronat, H. Stokking, R. van Brandenburg, "Inter-destination multimedia synchronization: schemes, use cases and standardization", Multimedia Systems, **I.F. (2011): 0.729**, 18(6), pp. 459-482, November 2012.

[Mon13a] **M. Montagud**, F. Boronat, H. Stokking, "Design and Simulation of a Distributed Control Scheme for Inter-Destination Media Synchronization", The 27th IEEE International Conference on Advanced Information Networking and Applications (AINA-2013), Barcelona (Spain), March 2013.

[Mon13b] **M. Montagud**, F. Boronat, H. Stokking, "Early Event-Driven (EED) RTCP Feedback for Rapid IDMS", The 21st ACM International Conference on Multimedia (ACM MM 2013), Barcelona (Spain), October 2013.

[Mon13c] **M. Montagud**, "Design, Development and Evaluation of an Adaptive and Standardized RTP/RTCP-based IDMS Solution", The 21st ACM International Conference on Multimedia (ACM MM 2013), Doctoral Symposium, Barcelona (Spain), October 2013.

[Mon13d] **M. Montagud**, F. Boronat, "RTP/RTCP and Media Sync: A Review and Discussion of Future Work", MediaSync Workshop 2013, Nantes (France), October 2013.

[Mon14a] **M. Montagud**, F. Boronat, P. Cesar, "A customizable open-source framework for measuring and equalizing e2e delays in shared video watching", ACM International Conference on Interactive Experiences for Television and Online Video (ACM TVX 2014), Newcastle (UK), June 2014.

[Mon14b] **M. Montagud**, "Design, Development and Evaluation of an Adaptive and Standardized RTP/RTCP-based IDMS Solution", Poster, I Encuentro de Estudiantes de Doctorado de la UPV, Universitat Politècnica de València (UPV), Valencia (Spain), June 2014, Link to poster.

[Mon14c] **M. Montagud**, F. Boronat, H. Stokking, P. Cesar "Design, Development and Assessment of Control Schemes for IDMS in a Standardized RTCP-based Solution", Computer Networks, **I.F. (2012): 1.231**, 70(9), pp. 240-259, September 2014.

[Mon15] **M. Montagud**, F. Boronat, H. Stokking, "Early Event-Driven (EED) RTCP Feedback for Rapid IDMS", https://tools.ietf.org/html/draft-montagud-avtcore-eed-rtcp-idms-00, Internet Draft, February 2015.

[Sto12] H. Stokking, R. van Brandenburg, F. Boronat, **M. Montagud**, "Standardization of Inter-Destination Media Synchronization", MediaSync Workshop 2012, Berlin (Germany), October 2012.

[Sto14] H. Stokking, O. van Deventer, F. Boronat, **M. Montagud**, "Inter-Destination Media Synchronizaton for IPTV Environments", IETF Internet Draft, Audio Visual Transporte CORE (AVTCORE) Group, draft-stokking-avtcore-idms-for-iptv-00, October 2014. Link to the slides of the presentation at the 91th IETF meeting.

Publications Under Review

M. Montagud, F. Boronat, B. Roig, A. Sapena, "How to Perform AMP? Cubic Adjustments for Improving the Smoothness of the Playout Curve", IEEE Communications Letters.

M. Montagud, F. Boronat, H. Stokking, P. Cesar "RTP/RTCP for Media Synchronization: A Review and Discussion of Potential Extensions", ACM TOMM.

J. Belda, **M. Montagud**, F. Boronat, F.J., Pastor, M. Martinez, "iWebSync: Towards a Web-based Platform for Distributed Media Synchronization and Social Interaction", IEEE ICME 2015.